AFRL-IF-RS-TR-2006-164
Final Technical Report
May 2006

JOINT EXPERIMENT ON SCALABLE PARALLEL PROCESSORS (JESPP) PARALLEL DATA MANAGEMENT

University of Southern California/ISI

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 074-0188*

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** MAY 2006	**3. REPORT TYPE AND DATES COVERED** Final Mar 2005 – Sep 2005

4. TITLE AND SUBTITLE JOINT EXPERIMENT ON SCALABLE PARALLEL PROCESSORS (JESPP) PARALLEL DATA MANAGEMENT	**5. FUNDING NUMBERS** C - FA8750-05-2-0061 PE - N/A PR - JESP TA - PP WU - DM
6. AUTHOR(S) Dan M. Davis, Robert F. Lucas	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Southern California/ISI 4676 Admiralty Way, Suite 1001 Marina del Ray California 90292-6601	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/IFTC 525 Brooks Road Rome New York 13441-4505	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER** AFRL-IF-RS-TR-2006-164

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Duane Gilmour/IFTC/(315) 330-3550 Duane Gilmour@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 Words)*

The need to present quantifiable results from simulations to support transformational finds is driving the creation of very large and geographically dispersed data collections. The US Joint Forces Command is conducting a series of Urban Resolve experiments to investigate concepts for applying future technologies to join urban warfare. The recently concluded experiments utilized and integrated multiple Scalable Parallel Processors (SPP) sites distributed across the United States. This computational power is required to model futuristic sensor technology and the complexity of urban environments. The Urban Resolve simulation generated more than two terabytes of raw data at a rate of >10 gigabytes per hour. The size and distributed nature of this type of data collection pose significant challenges in developing the corresponding data-intensive applications that manage and analyze them. We present here a next generation data management and analysis tool, called Simulation Data Grid (SDG). The design principles driving the design of SDG are: 1) minimize network communication overhead by storing data ear the point of generation and only selectively propagating the data as needed, and 2) maximize the use of SPP computational resources and storage by distributing analyses across SPP sites to reduce, filter and aggregate the data.

14. SUBJECT TERMS Scalable Parallel Processor Supercomputing, Data Management, Data Collection, Large Scale Simulation, Semi-Automated Force simulation	**15. NUMBER OF PAGES** 68
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

List of Figures

List of Tables

1.0 Executive Summary

JESPP-PDM (Joint Experiment on Scalable Parallel Processors: Parallel Data Management) explores the design issue in using SPP machines to build the next generation data management tools for processing and analyzing high-volume distributed data. This tool was called Scalable Data Grid, or SDG. In this report, an initial prototype implementation of this system is described. This project follows on earlier research, primarily reported in JESPP 02-04 Final Report for F30602-02-C-0213 (ISI, 2006). Interested parties will also want to read input from the High Energy Physics community in this area (Bunn, 2006).

Advanced data analysis techniques were both necessitated and enabled by leaps in computing power (ISI, 2006). The Joint Forces Command (JFCOM) JESPP project had effectively implemented High Performance Computing concepts for the joint semi-automated forces (JSAF) simulation. The challenges facing the defense analyst had grown to include the need to consider interactions of combatants and non-combatants, as well as a flood of other data. These requirements stretched both existing computational techniques and data analysis methodologies. The JESPP-PDM effort made progress on or achieved eleven tasks:

- Reviewed, improved and achieved consensus on the data management (DM) design
- Implemented the consensus design in an operational setting
- Provided for sufficient storage of data
- Reduced delays in processing raw data into database
- Optimized organization of data to facilitate analysis
- Implemented analytical tools to more fully exploit data
- Identified and enabled more sophisticated statistical analyses
- Enabled data mining
- Designed and enabled improved data visualization techniques
- Facilitated the situation awareness object (SAO) creation, management and display
- Enabled feedback of real-time situational data from entities.

The benefit to the end-users will continue to be the ability to review widely dispersed data from very large and costly simulation experiments and to enable more sophisticated analysis of the situation, producing extraction of more supportable insights.

2.0 Methods, Assumptions and Procedures

Previously, within the JESPP project, the communication infrastructure framework that enables very large-scale distributed simulation with high entity counts was developed. The Urban Resolve Phase I exercise utilized more than 100,000 entities using hundreds of compute nodes distributed across multiple geographical sites. A typical two-week Urban Resolve exercise event generated approximately 2 terabytes of raw message data. For this exercise, a distributed logging system that was capable of logging all of the message data was implemented. However, in terms of data management and data analysis, this distributed logging system had some limitations. These limitations will be described in this report, and how the SDG addresses these limitations.

3.0 Approach

The Scalable Data Grid is a distributed data management application/middleware that helps people deal with very large, geographically dispersed data sets over heterogeneous environments. The design principles driving the design of the SDG are 1) minimize network communication overhead (especially across SPPs) by storing data near the point of generation and only selectively propagating the data as needed, and 2) maximize the use of SPP computational resources and storage by distributing analyses across SPP sites to reduce, filter and aggregate the data.

The design goal of the SDG is to provide the data handling capabilities that are essential to current and future simulation analysis needs of the JFCOM. It should be capable of collecting and storing high volume/high rate data from geographically distributed data sources, browsing high-level summaries and overviews of the stored data, querying details of the stored data, and discovering what part of the data has changed.

These capabilities are applicable to multiple domains where large amounts of data are generated. In addition to distributed event-based simulations, such as JSAF, the SDG should also be useful to other data intensive applications, such as live instrumented exercises and instrumented physics experiments.

The data collected and the analysis tools provided are of potential use to a variety of people working with the simulation. To the military analysts, the logged data can be used to compute effectiveness measures, such as situation awareness. They can compare and contrast simulation ground truth against sensor observations. The simulation developers can use the same logged data for validation and verification. They can query the logs to check simulation events/patterns against expected behavior to find anomalous behavior. The logging can easily be adapted to log resource usage, such as CPU, memory and network usage. Infrastructure managers can use these data to discover faults and resource usage bottlenecks.

The initial performance goal of SDG was to be able to support JSAF simulations running with one million entities. Such high entity counts would generate, to within an order of magnitude, about 100 GB of data per hour. Over a typical two-week event, about eight terabytes of data would need to be collected. Large-scale JSAF simulations are typically distributed across multiple geographically dispersed sites. In the Urban Resolve experiments, the simulation was distributed across two supercomputers and multiple workstations at different sites. The sites included the Maui High Performance Computing Center (MHPCC), the Aeronautical Systems Center (ASC), JFCOM J9, the Space and Naval Warfare Systems Command (SPAWAR), and the Topographic Engineering Center (TEC). Figure 1 gives a visual overview that will be explicated in the following text.

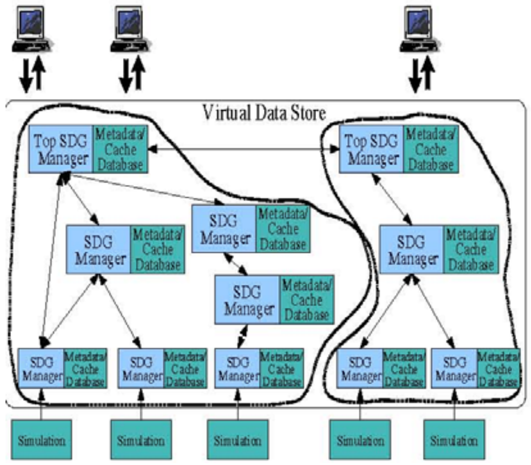


Figure 1. Conceptual Model, Dashed Loops Indicate the Boundary of Local Networks

3.1 Goals

Specific tasks that were investigated during the contract period of performance were:

- Review, improve, and achieve consensus on the DM design
- Implement the consensus design in an operational setting
- Provide for sufficient storage of data
- Reduce delays in processing raw data into a database
- Optimize organization of data to facilitate analysis
- Implement analytical tools to fully exploit data
- Identify and employ more sophisticated statistical analyses
- Enable and employ data mining
- Design and implement improved data visualization techniques
- Facilitate SAO creation, management and display
- Enable feedback of real-time situational data from entities

3.2 Background

JFCOM J9 and the Joint Advanced Warfighter Program (JAWP), staged several training and integration exercises in early 2004, followed by four experiments, each two weeks long, from June through October. Several sites participated in the events. The TEC site at Fort Belvoir, Virginia, had 30+ workstations and Saber, a quad-CPU machine with four terabytes of disk space that were used for after event storage. The SPAWAR site at San Diego, California, had 20+ workstations. The J9 Distributed Continuous Experimentation Environment at Suffolk, Virginia, had 50+ workstations and a 16-node mini-cluster. ASC, at Wright Patterson Air Force Base at Dayton, Ohio, had the Glenn cluster with 128 dual CPU nodes. The MHPCC site at Maui, Hawaii, had the Koa cluster with 128 dual CPU nodes.

The experiments typically ran five days a week, ten hours a day. Simulators might run all night, but with little activity and usually with logging disabled. Depending on availability and requirements, one or both of Glenn and Koa were used. Up to two hundred thousand clutter entities were simulated on the large clusters. (In this simulation, civilian entities are termed clutter, in that they serve to mask military entities.) Several thousand non-clutter entities were simulated on the other sites. A single node on the large clusters simulated 1000-2000 clutter entities.

Data logging was performed in two modes, near real-time and after action. Real-time data was inserted into an *SQLite* database. A node simulating 1000 clutter items would generate an SQLite database of approximately 50 megabytes in an hour. The databases were deleted and reinitialized when they grew to over a gigabyte. If 100 nodes of the cluster were used for clutter simulators, approximately 5 gigabytes per hour of data was generated. For after action use, compressed binary data was stored in an archive directory. Binary compressed data is approximately 1/7th the size of the corresponding database. Each night, the archived data was transferred to Saber, and expanded and decoded into a single *MySQL* database.

Clutter data from the Glenn and Koa clusters was not entered into the Saber database, due to size limitations. Data from 100 nodes on Glenn for a ten-day event would have been close to a terabyte. Data from TEC, SPAWAR, J9 and the J9 mini-cluster for non-clutter entities were entered into the *MySQL* database. The Urban Resolve Phase I exercise generated about a terabyte of data in the *MySQL* database.

The nightly data transfer was about 15 gigabytes of compressed data. Network transfer rate to Saber was approximately ten megabits per second. Three or four hours were required to do the transfer. Decoding and indexing the data into the *MySQL* database took 12 hours if everything worked perfectly. Human error and other factors usually prevented a day's data from being entered into the database before the next day's event started. It was usually at least several days after an event before the complete after action database was ready on Saber.

The logging methodology used for the four exercises in 2004 was adequate. It was the first attempt at logging data from hundreds of processors distributed geographically around the country simulating thousands of non-clutter entities. SDG is intended to remove deficiencies in the 2004 methodology and upgrade what was essentially an experimental system into a production system. The design parameters for SDG specifically address the following list of deficiencies in the 2004 system:

1. Near real-time and after action data logging are implemented differently. Near real-time queries are restricted by the use of simple aggregators.
2. The use of a single database on Saber does not have the capacity to include clutter data from the Glenn and Koa clusters.
3. Data transfers, decoding and indexing are time consuming and error prone, delaying the availability of the database. A goal is to have the complete database kept up to date continuously.

4. Retrieval of data and database generation for multiple exercises is inconvenient.
5. Expansion to more compute nodes, more entities per compute node and more data per entity is impossible. Disk storage, compute power, and network bandwidth all impose serious limitations.
6. The system does not respond gracefully to hardware and network problems. Saber is a single point of failure that makes all data unavailable.
7. Complex queries that may be useful to analysts are slow or impossible.

Database queries used in Urban Resolve are generally summary in nature. They count how many events or entities (database rows) meet specified criteria. Complex join operations were rarely, if ever, used. Were it not for this constraint on the queries, an efficient distributed design would be more difficult.

3.3 Specific Activities over the Period of Performance

As described in the previous section, data collected in the current system from the simulation are distributed and replicated at multiple locations. There is one access mechanism to query the data during simulation runtime, and another when the simulation is over. From the user's perspective, these data access complexities are unnecessary.

The SDG adds a data access middleware layer that hides these complexities and presents a simple coherent view of data to the user. From their desktops SDG users should be able to access and analyze the data without having to know:

1. How to access the data and what the network interconnection topology is (access transparency).
2. Where the data is located (location transparency).
3. Whether the data source has moved (migration transparency).
4. Whether the data is from a replicated source (replication transparency).
5. Whether data sources are shared (concurrency transparency).

Users interact with SDG through one of SDG's top-level managers. Users submit queries to a top-level manager, and they receive query results from a top-level manager. The SDG is capable of handling static data sets (no new data added), as well as dynamic data sets (new data continuously being added). For dynamic data sets, users can register static queries with top-level managers, and receive asynchronous query results. The SDG provides feedback to the users regarding the queries they submit, by letting them know the resources required to execute the query and the resources currently available.

4.0 Results and Discussion

4.1 Review, Improve and Achieve Consensus on the DM Design

Over the period of performance of this contract, several meetings were conducted with JFCOM personnel, other DoD analysts, representatives of the High Performance Computing Program and other high performance data management colleagues. The announced intent of these meetings was the establishment of a consensus on the goals of the logging, manipulation, storage, organization and retrieval of the data generated by large-scale simulations conducted by the Joint Forces Command.

Some practical considerations were achieved and many new avenues of investigation were discovered. The necessity for the system architects' articulating their capabilities to stimulate the vision of the simulators and analysts was one major insight from this activity. Having in the past been relegated to reviewing rather prosaic data, it is not easy for simulation professionals and analytical scientists to envision what new techniques could contribute to their mission. The JESPP-PDM process was one that helped stimulate thinking that is just now appearing in the literature and will be implemented in future generations of PDM systems.

A consensus emerged very rapidly that was centered on leaving the data in a distributed configuration, saving bandwidth and providing fault tolerance by not storing the data at some arbitrary central location, and developing a distributed method of retrieving data when needed by remotely located analysts.

These broad concepts then drove the SDG conceptual design process. This has resulted in the implementation as reported herein. Both the analyst and the simulator communities were fully supportive of this approach.

4.2 Implement the Consensus Design in an Operational Setting

Since its inception, the design has been fully implemented on the JFCOM simulation network, including nodes at JFCOM (Suffolk VA), TEC (Ft Belvoir VA), the Institute for Defense Analysis (IDA) (Alexandria VA), ASC-MSRC (Wright Patterson AFB OH), SPAWAR (San Diego CA) and MHPCC (Maui HI). The scalability of the implemented design will allow addition of more nodes in the future without significant degradation of either simulation performance, data management performance or data analysis performance.

This implementation was accomplished using *SQLite* and *MySQL*, with further programming in the JAVA programming language. Constant upgrades continued during the period of performance of this project. This upgrade and enhancement activity is anticipated to be on-going for the life of the JESPP project.

Further implementation is planned for 64 bit processors such as the Advanced Micro Devices (AMD) Opteron. Also, a dual node Opteron test bed was procured and assembled at the

University of Southern California's Information Sciences Institute (ISI) for test. To date, these tests have shown the easy portability of the SDG code and the good performance characteristics on the AMD computer. At this point, insufficient data has been generated to correctly evaluate the efficacy of this implementation and its ability to scale at the level desirable by both the researchers and the users at JFCOM.

4.3 Provide for Sufficient Storage of Data

The physical storage necessary for the scale of simulations commonly conducted by JFCOM is problematic and continues to be of concern. Analysis has been conducted indicating that a typical simulation stores on the order of a few terabytes of data. The problem is that nearly 80% of the data is discarded. If all of the data were collected, 10 Terabytes would be expected. If the sensor data were stored, another 2X increase would result. Scaling to the 10 million entities desired by JFCOM would then produce a couple of orders of magnitude increase on those numbers. The net result is the necessity of providing huge amounts of storage somewhere in the system.

One option sought was the provision of extra storage by the High Performance Computing Modernization Program (HPCMP), but this has not proven fruitful. The HPCMP is not currently funding storage facilities in their Distributed Center program. Instead, JFCOM has purchased additional storage at Suffolk, and ISI has been instrumental in foraging for additional storage at the remote sites.

4.4 Reduce Delays in Processing Raw Data into a Database

One method of reducing delays in processing is to utilize the distributed processors on the SPPs at Maui and in Ohio. This method has been effectively implemented utilizing the otherwise lightly-used second processors on the SPP nodes for processing the data. Performance measurements have not indicated any degradation of simulation performance, while the data processing was taking place simultaneously.

As with other issues, this will not always be the case. Scalability is a very tenuous proposition and any change in processor load, communications bandwidth demands or storage latencies may rapidly impact the earlier exceptional performance of the scalable systems. The SDG has taken these future goals into account while designing the system architecture and, at least so far, this has proven to be effective.

4.5 Optimize Organization of Data to Facilitate Analysis

Close coordination between the designers, programmers and simulation analysts has been maintained and is reflected in meetings, such as the one in March of 2005 in Marina del Rey to discuss analytical requirements. Consistent implementation has provided an early testbed for the analysts to deliver feedback to the designers and programmers concerning the utility of the data system in the analytical process. Real-time access was designed, implemented and tested and is one example of the new capabilities delivered to the analysts.

This real-time capability allows the analyst to provide feedback to the simulation operators, during the simulation, to facilitate achieving program goals. The multi-dimensional data design described below is also effective in providing the analyst with a rapid and logical way to access data for analysis and for further processing.

Future analytical techniques have been surveyed and the system will putatively support them when they are applied. Additional monitoring of performance, in all the domains of the system will be necessary to optimize the utility of the program.

4.6 Implement Analytical Tools to Fully Exploit Data

A number of analytical tools have already been implemented or enabled by the SDG architecture. Some of these were suggested and generated by the ISI team and some flowed from the IDA analytical personnel.

Select Queries. User sends a *select* Structured Query Language (SQL) query to a top-level manager. The top-level manager returns the query result in a result set table. The underlying data is stored in multiple locations, but to the user it appears to be one big centralized database. One can further classify select queries into aggregation queries, union queries and simple queries. Aggregation queries involve operators like sum, min, max, and average. Union queries access data from more than one table and/or results from sub-queries. Simple queries do not involve aggregation operators or unions.

Sample simple queries include: return all entity weapon damage reports within the last 30 minutes; and return red tank movements within the last 10 minutes. Sample union queries include: return all entities that were painted by a sensor; and return marking information of the entity that fired a weapon within the last 10 minutes. Sample aggregate queries include: count the sensor tracks grouped by sensor type, or group by degree of assuredness; return killer/victim scoreboards; and return sensor/target scoreboards

For simple select queries SDG managers only need to concatenate results returned by sub-tasks without further processing. The previous implementation supports simple queries. The next section describes our current effort to extend to aggregation queries and multidimensional analysis.

Resource Usage Explanation Queries. This is similar to *MySQL's* EXPLAIN command. Given a select query, SDG traces through the execution of the query, and explains which resources and how much were used to answer the query.

Canned Queries. User defines periodic/trigger select queries. Based on the defined period, or the trigger, SDG executes the query and asynchronously returns the result to the user. Sample uses include: receive alerts when a missile is launched; and automatically update killer-victim scoreboard when a weapon is fired.

The Globus' Data Access and Integration (DAI) service provides a common web services interface for accessing heterogeneous data sources (files, relational, extensible markup language (XML)). The relational part of the interface allows clients to submit SQL queries to data services and to receive query result sets from the data services. Also, DAI supports asynchronous delivery, which may be useful for periodic canned queries. In addition, Distributed Query Processing (DQP) service layers distribute join capabilities on top of DAI.

However, one key reservation about using DAI/DQP is the overhead of using simple object access protocol (SOAP)/XML based communication for query result processing. Using standards-based communication makes sense if the data sources are heterogeneous. But, in this case, the focus was just on relational data. Furthermore, scalability to handle very large simulation data sets is one of our overriding concerns.

4.7 Identify and Employ More Sophisticated Statistical Analyses

Discussions were held with the High Energy Physics researchers at Caltech and the Evolutionary Computing visionaries at Natural Selection in San Diego to consider how the data presented might better be utilized to enhance mission accomplishments for an entire range of DoD organizations who are currently using entity-level simulation. Each module in the SDG was subsequently evaluated to assure the architects and programmers that the requisite capabilities were being built into the code to support more sophisticated statistical analyses in the future.

Some of the concepts appearing in the financial world have also been reviewed for applicability to the simulations conducted by JFCOM. Particularly, some of the work focusing on the Path Integral concepts of Richard Feynman of Caltech has been implemented by ISI colleagues and JESPP team members. They can establish the sensitivity of final results in a Monte Carlo simulation to the input parameters. The SDG system would facilitate the inclusion of this tool in the future, should that prove propitious.

4.8 Enable and Employ Data Mining

Early work on data mining was directed toward not doing anything that would inhibit later implementation of this technique. All data structure designs were developed with future data mining possibilities in mind. Our knowledge of data mining approaches increased when one of the principal researchers on the project was asked to teach at course at the University of Southern California (USC) on the subject. The background investigation performed while preparing for the course increased our awareness of data mining research and applications, and helped us identify research areas to pursue in support of this project.

Data mining will require the scanning of all of the distributed data that is germane to that particular issue, unconstrained by preconceived notions as to which data are most important. The SDG data structures will support this trans-continental data analysis.

4.9 Design and Implement Improved Data Visualization Techniques

One of the burgeoning fields high performance computing is assisting the analyst is in perceiving trends and correlations in what might otherwise seem to be chaotic data. Several utilities have been included in the SDG to enable that process. The data and the system are made manageable by careful design of the system control functions.

System administrative functions are used to manage the SDG system itself. The functions needed include the ability to remotely manage and control startup and shutdown of managers, the ability to remotely monitor the health of managers, and the ability to map task decomposition hierarchies and data flow diagrams onto the managers.

Further, the system was designed with the concepts of visualization of data in the future being a primary focus. Current analysts conceive visualization only as representing images of ongoing battles in formats approaching photo realistic, so there is an education process required to get users to consider the possibilities of insights from properly visualized data, insights that otherwise would be lost.

These system administration functionalities match well with Globus' Execution Management components and Information Services' Monitoring and Discovery System (MDS). MDS's Index Service is able to register services, as well as maintain resource properties associated with the service.

4.10 Facilitate SAO Creation, Management and Display

Currently, JSAF records player’s situation awareness by having them annotate “Situation Awareness Objects” (SAOs) on the computer screen during the exercise. SAOs are pointers that indicate the presence and direction of movement of the opposing force. When the exercise is complete, overall SA is evaluated by comparing the total SAOs recorded against the opposing forces activities. As it stands now, the process is manual and subjective, which leaves room for improvement.

The design of the SDG has been carefully configured to allow better SAO management in the future. Situation awareness will almost surely be one of the major thrusts for JFCOM in the years to come. Again, close coordination with the IDA and TEC personnel involved in SAO analysis drove the design of the SDG so as to assist them in SAO data management.

Subjective evaluation or measurement based on verbal protocols or interviews that are given after the exercise can include: (a) an effectiveness form, (b) individual or collective group discussion or hot wash, and (c) a third party observer. The SDG has been implemented in a way that supports any of these three scenarios.

4.11 Enable Feedback of Real-Time Situational Data from Entities

This was accomplished by substituting real-time conversion and processing for the previous batch processing that took place at night, after the analysts had secured for the day. The new system performs processing in real-time, at the remote computing site and therefore makes it possible to deliver the data to the analyst in a real-time manner. Careful programming precludes performance degradation, yet provides an entirely new capability to the analysts.

While Figure 2 shows some single point user interfaces, in practice the users are similarly distributed trans-continentally and the system is able to accept queries and return data to any node on the net. The net is now constrained by classification, not by system design.

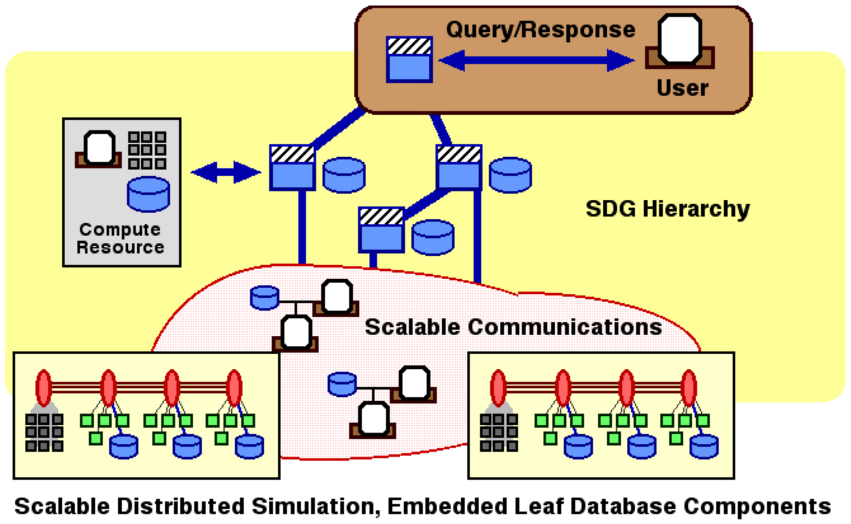


Figure 2. Diagram Showing Real-Time Distributed Data Processing

5.0 Activity Discussion

The JESPP-PDM project has delved into some of the *conundra* of data management science. This report now surveys some of this research.

In 1991, Peter Deutsch articulated eight fallacies of distributed computing (Deutsch, 2005):

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. The topology doesn't change
6. There is one administrator

7. The transport cost is zero
8. The network is homogeneous

Distributed software systems developed under these assumptions tend to be brittle. They later have to be re-engineered to work around these assumptions.

Potential additional fallacies related to distributed data computing are:

1. Disk capacity is infinite
2. Disk latency is zero
3. Disk bandwidth is infinite
4. Data processing cost is zero

The initial design of the system was based on the assumption that there is sufficient local disk space to store the logged data. But, it was found that nodes on supercomputer clusters tend to have less local disk space than the average desktop computer. First it was necessary to implement a near-real-time system, then an off-site after action post-processing system.

SDG manages these potential pitfalls through a unique division of labor. SDG hides some of the networking details from the user by explicitly managing the five transparencies listed in Section 3.3, but exposes other details (e.g., resource usage and storage options) to allow users to examine, and if needed, override default behavior and manage those details themselves.

To address the distributed data computing fallacies, it is intended to provide multiple data services with varying levels of capabilities to let the user select the appropriate services for the tasks at hand (Table 1). These are storage options that trade-off storage size, query speed and query preprocessing time. For the Urban Resolve exercises, approximately 50% of the messages were not logged because these messages were internal simulation bookkeeping messages. For example, Culture Intersection (CultureInt) messages that determine which car should enter an intersection next usually are of no interest to the analysts.

Typically, a user may want to include a compressed storage option to keep an archive of the simulation data. Then, the user may wish to select another storage option, for all or a partial subset of the messages, for faster querying. If the user chose multiple storage options, he has the option of deleting/truncating data storage to recover disk space.

Table 1. Range of Storage Options that Trade-Off Storage Size, Query Speed and Preprocessing Time

Storage Options	*Storage Size*	*Query Speed*	*Query pre-processing*
Do not log	***Zero***	***N/A***	***None***
Compressed (raw)	***Small***	***Very slow***	***Small***
Text (decoded)	***Medium***	***Slow***	***Small***
Database	***Medium***	***Medium***	***Medium***
DB w/ indexing	***Large***	***Fast***	***Large***
Cube (D = # of dimensions)	***Large, for high D***	***Fast***	***Large, for high D***

5.1 Conceptual Model

SDG managers perform all of the data access/query/management tasks. Conceptually, there are three types of managers: top-level, data source and worker.

Top-level managers have published addresses. Users connect through the top-level managers. To minimize network traffic, typically there is at least one top-level manager for each local area network. Top-level managers know how to connect to each other. Non-top-level managers know how to connect to at least one top-level manager (Figure 1).

Data source managers store the actual data. Other applications insert data into data source managers through defined Application Programming Interfaces (APIs).

Worker managers perform most of the work within the system. When given a data processing task, the top-level manager decomposes a task into sub-tasks. Depending on the nature of the task, the top-level manager enlists one or more manager workers, data source or other top-level managers. It then assigns sub-tasks to these managers and data source managers. Finally, it defines a data flow topology linking together the sub-task executions.

The mapping of the tasks onto managers must take into account and take advantage of a heterogeneous computing environment. The networking infrastructure within a local cluster typically uses gigabit Ethernet, or even faster proprietary Myrinet. The inter-cluster networking infrastructure is typically orders of magnitude slower. Computation must be moved closer to the data sources to avoid transportation penalties.

The storage hierarchy varies from cluster to cluster. For example, the original configuration of the Koa cluster at the Maui Supercomputing Center does not include local hard drives. A ten terabyte storage area network (SAN) mount on a global file system (GFS) functions as the only secondary storage. The Glenn cluster at ASC has a total of ten terabyte storage mounted on local hard drives, and ten terabyte SAN storage mounted on GFS.

5.2 Data Administrators

Data administrative functions are used to manage the data collected and stored within the system.

Monitor data/resource usage statistics. Monitor the rate and size of data flowing into the SDG system, monitor the available disk capacity, monitor network usage, and monitor CPU usage.

Archive data sources. Copy/move data sources into one centralized location. This is useful for archiving data into a centralized SAN or a tape archival system.

Merge/split data sources. Combine multiple data sources into one source. Partition one data source into multiple data sources. These operations are useful to take advantage of parallelism when extra compute resources are available.

MDS is able to interface with cluster monitoring tools, such as Ganglia (Ganglia, 2005), to produce up-to-date system load/usage information. In addition, GridFTP, Reliable File Transfer service and Replica Location Service also play important roles.

5.3 Distributed Multi-Dimensional Analysis: Sensor/Target Scoreboards

One of the key focus areas of Urban Resolve Phase I was to study the effectiveness of future Intelligence, Surveillance and Reconnaissance (ISR) sensors in helping soldiers operate in complex urban environments. The Sensor/Target (S/T) scoreboard provides a visual way of quickly comparing the relative effectiveness of individual sensor platforms and sensor modes against different types of targets. The S/T Scoreboard is a specific instance of the more general multidimensional analysis.

In the Urban Resolve federation, a simulated sensor entity lays down sensor footprints to delimit sensor coverage sweep. For each target entity within the footprint, a *contact report* is generated to hold the result of the sensor detection. The contact report includes information about the sensor entity, the platform the sensor entity is mounted on, the sensor mode, the target entity, the detection status, the perceived target type, the perceived target location, the perceived target velocity and so on.

Sensor/Target scoreboards have the capability of providing summary views by aggregating individual sensor platforms into sensor platform types; such as *high altitude*, *medium altitude*, and *low altitude*. And, it aggregates individual target entity objects into target classes, which can range from the generic (like *Civilian Large Trucks*) to the specific (like Russian *MAZ-543 MEL*). As described by Graebener (Graebener, 2003), the current implementation of the scoreboard provides four levels of details. The information provided are:

1. Table of contact report counts broken down by sensor platform types and by target classes.
2. Given a sensor platform type and a target class, table of number of contact report counts broken down by sensor platforms and by sensor mode.
3. Given a sensor platform and a sensor mode, list of target objects.
4. Given a target object, list detailed target object attributes.

Initially, the S/T scoreboard displays the level one aggregate table of sensor platform types and target classes. By clicking on a table cell (i.e., specifying a particular platform type and target class), the S/T scoreboard brings up the level two display of sensor platforms and sensor modes. Sensor modes are methods of detection, such as Moving Target Indicator (MTI) and Synthetic Aperture radar (SAR) Spot and SAR strip.

5.4 Analysis of S/T Scoreboard from Multidimensional Perspective

The current implementation of S/T scoreboards projects the contact reports along three dimensions for analysis. The three dimensions are sensor platform, target object and sensor mode. In addition, sensor platforms are aggregated into sensor platform types, and target objects are aggregated into target classes. Figure 3 depicts these three dimensions as linear partial orderings. These dimensions can be crossed to create lattices, as shown in Figure 4.

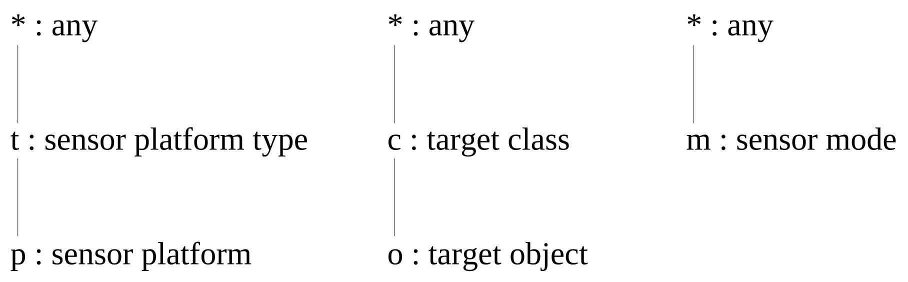


Figure 3. Three Possible Dimensions to Partition the Data for Analysis

With respect to the right lattice in Figure 4, the information contained in the level-one S/T scoreboard corresponds to the node *tc*, sensor platform type by target class. The level-two information corresponds to slices of node *pcm*, where *p* is restricted to a particular sensor type, *t*, and entity class, *c*. Levels three and four correspond to target objects specified by cells in node *pom*.

The four levels of the S/T scoreboard present information useful to the analysts. But, other nodes within the lattices may be of potential interest. For example, node *cm* summarizes the effectiveness of sensor modes with respect to target class. Or, node *pc* summarizes the effectiveness of sensor platforms with respect to target class. In addition, other dimensions not used in S/T scoreboards may be of potential interest, for example detection status, time, location, terrain classification (high-rises, low-rises, and flat), weather conditions, and so on.

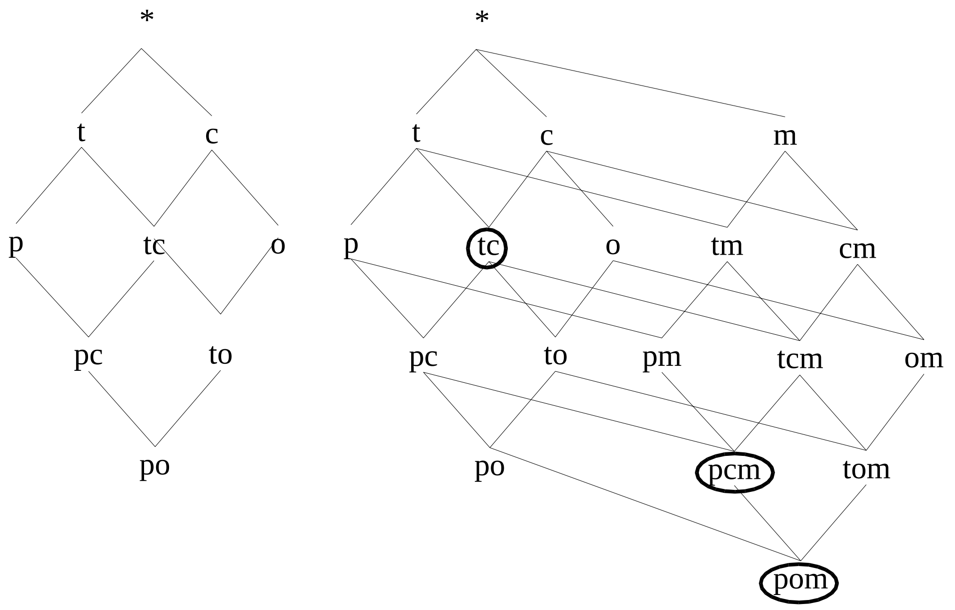


Figure 4. Lattices are Generated by Crossing the Dimensions. Crossing the Sensor Dimension with the Target Dimension Generates the Lattice on the Left. Crossing the Left Lattice with the Sensor Mode Dimension Generates the Lattice on the Right

5.5 Multidimensional Analysis

The S/T scoreboard falls into an analysis class called multidimensional analysis, or sometimes called On-Line Analytical Processing (OLAP) or data warehousing (Kimbal *et. al.*, 1998). Other types of scoreboards, like Killer/Victim and Truth/Perception, are also multidimensional in nature. Conceptually, the data structure used to store multidimensional analysis data is the *cube*. The two-dimensional array data structure used to store a two-dimensional scoreboard is extended to higher dimensions.

Users could query the OLAP system for the entire cube, but typically the users are more interested in projections and partial views of the cube. Operations on the cube include roll-up, drill-down and slice & dice. Roll-up aggregates data along a dimension to hide details. This corresponds to walking up the dimension lattice. Drill-down partitions the data along a dimension to reveal more details. This corresponds to walking down the dimension lattice. Slice & dice selects subsets of the cube elements.

5.6 Query and Data Characteristics

Query and data characteristics within the simulation differ from traditional OLAP assumptions in two significant ways: 1) a query is concurrent with insertions, and 2) the data is distributed. The first was driven by the JFCOM operational paradigm; the second was a function of the commitment to Network Centric Warfare and distributed redundancy and fault tolerance.

Typically, OLAP is performed on historical data. For example, retail chains may keep sales transaction records to determine their best performing stores, or emerging consumer trends. This analysis is usually performed off-line. The analysis need not be updated as individual sales transactions occur. This is one of the myriad ways in which the relevance of the scientific computing community was more relevant than the transaction processing environment that is so much the domain of the commercial data base companies like Oracle.

In addition, data is typically sent to a centralized facility to be analyzed. In our case, it is not feasible to centralize the data because of the amount of data and near-real-time nature of the query. Our data is logged locally at the point of generation. If there are 100 simulation nodes, then there are 100 local logs.

Previous works have studied distributed OLAP implementations (Goil and Choudhary, 2001; Beynon *et. al.*, 2002). Typically, they employ some type of data partitioning scheme to perform load balancing and/or to reduce input/output (I/O) overhead. For example, in the *chunking* data partitioning scheme, the data cube is partitioned into smaller sub-cubes. The number of dimensions of the sub-cubes remains the same, but now each dimension holds just a subset of the possible dimension values.

In our case these data partitioning schemes are not applicable. The simulation setup and placement dictate our data partitioning scheme. Moving these messages creates network traffic that may disrupt the actual simulation. Since it is not possible to preposition data, an alternative is investigating cube compression techniques to minimize storage and the I/O needed to aggregate the local cubes. These techniques include partial cube materialization (Harinarayan, 1996) that selectively pre-computes a subset of lattice nodes, coalesced cubes (Sismanis and Roussopoulos, 2004; Sismanis et al., 2002), and shell fragments (Li et al., 2004) that offer compact ways of storing the cube.

5.7 Implementation Status

OLAP systems on single processors are widely used and described in the literature. Two implementations are frequently used. Multidimensional On-Line Analytical Processing (MOLAP) stores multidimensional data in an explicit multidimensional structure. Relational On-Line Analytical Processing (ROLAP) stores multidimensional data in a relational database. MOLAP provides faster access to data, but ROLAP stores sparse data more efficiently. ROLAP was chosen for the implementation of SDG for two reasons. First, the ability to scale to very large data sets with potentially high number of dimensions was desired. ROLAP implementations tend to provide better scaling with respect to storage. Second, the current logger is implemented on top of relational databases. It was desirable to maintain backward compatibility to allow the analysts to use SQL to directly query underlying logged data.

It was planned to develop the system and implement features in an incremental fashion in order to deliver capabilities to J9 in a reasonable fashion. This has the added benefit of providing feedback, which can be applied to future development. The sensor target scoreboard was identified, as was discussed earlier, and it has critical features that are representative of many key features that would ultimately be required, and could be implemented quickly and efficiently.

One week of archived data from one Urban Resolve event was chosen as the test data. The sensor target scoreboard was prepared from a table in the database named I_ContactReport. There was interest in deriving a unique value for the type of sensor, the type of target and the detection status for each row of the table. This information is used to create a three-dimensional table of counts for each unique combination. Other information is discarded at this time. Future enhancements will incorporate information, such as time and location, to create a five-dimensional table (or larger).

The I_ContactReport table from our test case has approximately 18 million records. One column, node, identifies the machine on which the row was generated. To simulate the distributed generation of the data, four new databases were created based on applying a regular expression to the value in the node column. Only 16 columns were copied to the four new databases. Two additional columns were added to identify unique combinations of the target and of the sensor.

Next, a procedure was applied to each of the four new databases. In a real exercise, this procedure would be applied independently and concurrently on each computer maintaining a database.

The procedure consists of the following steps using *MySQL* commands:

1. Create a table of unique combinations of sensor values and unique combinations of target values. Assign an enumerated type to each.
2. Create a row in the table for each combination of sensor type, target type and detection status that occurs in the I_ContactReport table. Compute a column count giving the number of times the combination occurs. With appropriate indexing, this takes six minutes for six million records in one of the four sub-databases.
3. Add rows to the table for "wildcards" as appropriate for a data cube. A row is created for any sensor, any target, any detection status and three wildcards. This should equal the number of rows in the contact table for a three-dimensional table. Do the same for all permissible use of two wildcards and one wildcard.

This procedure is applied when a new dataset is introduced to the system. It is then applied to any new data that is added to the system. The data cube is always up to date.

There are now four relatively small databases containing complete and nearly instantaneously accessible information on the count of any combination of sensor types, target types and detection types. A user query to a top-level data manager is relayed to low level data managers connected to each of the four sub-databases. The responses are merged by combining responses with the same dimension value and summing the count field. The result is returned to the user.

6.0 Conclusions

The use of large clusters (hundreds of nodes or more) of processors to meet the demand for high performance computing is becoming a mature technology. The extension to use multiple clusters is likewise common, but less mature. The use of OLAP technology to analyze large data sets also is becoming a mature technology. To support JFCOM and JAWP, distributed clusters and OLAP were combined. Using this approach and innovation in key areas, it is possible to support JFCOM's current and expanding needs. A key principle is to store data close to its source to minimize network traffic. A second principle is to utilize the computational and storage resources of distributed clusters for database functions. These two principles reinforce, rather than interfere with, each other in the design and implementation of the data grid. A key area of innovation is the intelligent manager. The intelligent manager categorizes a query, creates an execution plan, distributes the work for the query, aggregates and delivers the results. The queries required and commonly used by JSAF analysts are efficiently executed by this system. Fault tolerance and realistic data archiving are additional benefits of our implementation. The system will be maintained and extended to include more processors, more clusters, larger datasets and more robust queries as required by DoD users.

Appendix A - References

- Barbara, D., Sullivan, M. (1998). “Quasi-Cubes: A space-efficient way to support approximate multidimensional databases”, Technical report, ISE Dept., George Mason University.
- Beynon, M., Chang, C., Catalyurek, U., Kurc, T., Sussman, A., Andrade, H., Ferreira, R., Saltz, J. (2002). “Processing large-scale multi-dimensional data in parallel and distributed environments”, *Parallel Computing.*
- Bunn, J.J. & Gottschalk, T.D., (2006), “Incorporating High Energy Physics Data Capabilities into Large-Scale DoD Simulations”, Interservice/Industry Training Simulation and Education Conference, Orlando Florida (Publication Pending).
- Deutsch, P. (2005). “The Eight Fallacies of Distributed Computing”, http://today.java.net/jag/Fallacies.html.
- Open Grid Services Architecture for Distributed Systems Integration. *Open Grid Service Infrastructure WG*, *Global Grid Forum*, June 22, 2002.
- Ganglia (2005). http://ganglia.sourceforge.net/
- Graebener, R.J., Rafuse, G., Miller, R. & Yao, K.T. (2003). “Successful Joint Experimentation Starts at the Data Collection Trail”. I/ITSEC 2003.
- Graebener, R.J., Rafuse, G., Miller, R. & Yao, K.T. (2004). “Successful Joint Experimentation Starts at the Data Collection Trail—Part II”. I/ITSEC 2004.
- Goil, S., & Choudhary, A. (2001). PARSIMONY: “An Infrastructure for Parallel Multidimensional Analysis and Data Mining”, *Journal of Parallel and Distributed Computing*, v61.
- Harinaryan, V., Rajaraman, A. & Ullman J.D. (1996). “Implementing Data Cubes Efficiently”. Proc. ACM SIGMOD '96, p 205-216.
- ISI Final Report, (2006), Contractor’s Final Report, “Joint Experimentation on Scalable Parallel Computers”, 02-04, Air Force Research Laboratory, Rome Labs, F30602-02-C-0213, September 2002 through September 2004.
- Pedersen, T. B., & Jensen, C. S. (2001) “Multidimensional Database Technology”. IEEE, Dec 2001.
- Riedewald, M., Divyakant, A., El Abbadi, A. (2001). “Flexible Data Cubes for Online Aggregation”. *Proceedings of the 8th International Conference on Database Theory*.
- Tenenbaum, A. S. & van Steen, M. (2002). *Distributed Systems principles and paradigms*, New Jersey: Prentice Hall.

Appendix B

Simulation Data Grid: Joint Experimentation Data Management and Analysis

Ke-Thia Yao and Gene Wagenbreth
Information Sciences Institute
University of Southern California
Marina del Rey, California
kyao@isi.edu, genew@isi.edu

ABSTRACT

The need to present quantifiable results from simulations to support transformational findings is driving the creation of very large and geographically dispersed data collections. The Joint Experimentation Directorate (J9) of USJFCOM and the Joint Advanced Warfighting Project is conducting a series of Urban Resolve experiments to investigate concepts for applying future technologies to joint urban warfare. The recently concluded phase I of the experiment utilized and integrated multiple scalable parallel processors (SPP) sites distributed across the United States from supercomputing centers at Maui and at Wright-Patterson to J9 at Norfolk, Virginia. This computational power is required to model futuristic sensor technology and the complexity of urban environments. For phase I the simulation generated more than two terabytes of raw data at rate of >10GB per hour. The size and distributed nature of this type of data collection pose significant challenges in developing the corresponding data-intensive applications that manage and analyze them.

Building on lessons learned in developing data management tools for Urban Resolve, we present our next generation data management and analysis tool, called Simulation Data Grid (SDG). The design principles driving the design of SDG are 1) minimize network communication overhead (especially across SPPs) by storing data near the point of generation and only selectively propagating the data as needed, and 2) maximize the use of SPP computational resources and storage by distributing analyses across SPP sites to reduce, filter and aggregate. Our key implementation principle is to leverage existing open standards and infrastructure from Grid Computing. We show how our services interface and build on top of Open Grid Services Architecture standard and existing toolkits (Globus). SDG services include distributed data query/analysis, data cataloging, and data gathering/slicing/distribution. We envision SDG to be a general-purpose tool useful for a range of simulation domains.

ABOUT THE AUTHORS

Ke-Thia Yao is a research scientist in the Distributed Scalable Systems Division of the University of Southern California Information Sciences Institute. Currently, he is working on the JESPP project, which has the goal of supporting very large-scale distributed military simulation involving millions of entities. Within the JESPP project he is developing a suite of monitoring/logging/analysis tools to help users better understand the computational and behavioral properties of large-scale simulations. He received his B.S. degree in EECS from UC Berkeley, and his M.S. and Ph.D. degrees in Computer Science from Rutgers University. For his Ph.D. thesis he implemented a spatial and physical reasoning system that automatically generated grids for novel geometries for computational fluid dynamics simulators.

Gene Wagenbreth is a Systems Analyst for Parallel Processing at the Information Sciences Institute at the University of Southern California, doing research in the Computational Sciences Division. Prior positions have included Vice President and Chief Architect of Applied Parallel Research and Lead Programmer of Pacific Sierra Research, where he specialized in tools for distributed and shared memory parallelization of Fortran programs. He has also been active in benchmarking, optimization and porting of software for private industry and government labs. He has programmed on CRAY, SGI, Hitachi, Fujitsu, NEC, networked PCs, networked workstations, IBM SP2, as well as conventional machines. He received a BS in Math/Computer Science from the University of Illinois in 1971.

Ke-Thia Yao and Gene Wagenbreth
Information Sciences Institute
University of Southern California
Marina del Rey, California
kyao@isi.edu, genew@isi.edu

INTRODUCTION

The specific motivation to develop the data logging and retrieval system described in this paper is to support simulations by the Joint Forces Command using JSAF (Joint Semi Autonomous Forces) software. JSAF provides entity-level simulation of ground, air and naval forces. Simulation of civilian entities is performed by a separate program, clutter. Simulation of multiple sensor platforms is performed by a program called SLAMEM. JSAF scales from a single cpu to hundreds of cpu's. Individual simulators run on a single cpu. The HLA publish/subscribe software architecture is used to communicate results between simulators. A software router network enables the system to scale to hundreds of processors.

In the last two years a requirement for a large increase in the number and the fidelity of simulated entities justified the upgrade of JSAF simulations from a network of workstations on a LAN, simulating a few hundred or thousand entities to a WAN including multiple Beowulf clusters and hundreds of processors simulating hundreds of thousands of entities.

An important part of the simulations is to log what happens for near real time and after action analysis. The broad range of analysis requires that nearly all data be logged. The mechanism used is to log data when it is published. The earliest implementation included in the simulation a software logger, which subscribed to, and received all data published anywhere in the simulation. The total size of the logged data was limited to 2 gigabytes. This worked when the number of processors and the number of simulated entities was small.

In 2003, to support larger simulations on Beowulf clusters ISI implemented a distributed logger. Data is logged locally on each processor running a simulator. Near real time data queries are supported by a simple tree system to broadcast a query and concatenate results. After action queries are supported by transferring compressed binary files to a single host and expanding into a single monolithic database.

In 2005, this implementation is no longer adequate. The current size of a database for a 2 week exercise, omitting nonessential data, is over a terabyte. The time required to transfer data to a single site and insert it into a database is inconvenient. Maintenance of hardware and software to support multiple large databases (one per exercise) on a single system is difficult. The current system can not support anticipated future increases in the size and fidelity of exercises and the amount of data to be logged.

This paper describes Simulation Data Grid (SDG), the data logging system designed and implemented to support large JSAF simulations. SDG utilizes the resources of the systems generating the data, distributed processors and storage. Logging resources thus scale as processors are added to support the simulation.

SIMULATION DATA GRID

Overview

Simulation Data Grid (SDG) is a distributed data management application/middleware that helps people deal with very large, geographically dispersed data sets over heterogeneous environments.

These capabilities, essential to current and future simulations by the Joint Forces Command, are provided by SDG:

- collect and store high volume/high rate data from geographically distributed data sources
- browse high summaries and overviews of the stored data
- query details of the stored data
- discover what part of the data has changed

- transport and redistribute the data

These capabilities are applicable to multiple domains where large amounts of data are generated, such as:

1. Distributed event-based simulation, e.g. JSAF
2. Live instrumented exercises
3. Instrumented physics experiments

Types of people that use these capabilities:

- Domain data analysts (e.g., analyze what happen during a simulation)
- Simulation developers (e.g., monitor simulation software behavior)
- System administrators (e.g., monitor resource usage)

The initial performance goal of SDG is to be able to support JSAF simulations running with 1 million entities.

- 100 GB of data per hour
- 8 Terabytes for a two-week event
- data distributed across two supercomputers and multiple sites
  - Maui High Performance Computing Center (MHPCC)
  - Aeronautical Systems Center (ASC)
  - US Joint Forces Command (JFCOM) J9
  - Space and Naval Warfare Systems Command (SPAWAR)
  - Topographic Engineering Center (TEC)

Risk of not developing SDG:

- losing valuable data from exercises/experiments
- being mired in inaccessible and unusable data
- producing false analyses and unsupported conclusions
- logger limiting the performance of the simulation

Leveraging Grid Computing

SDG is intended to operate in a joint experimentation environment, where the computing software and hardware elements may be quickly assembled on an as needed basis. The constituent elements may change depending on need and on resource availability. Each Urban Resolve exercise in a literal sense is setting up a virtual computing organization to solve a significant problem. This virtual organization spans multiple administrative domains each with its own security policies, and each offering its unique combination of computing, networking and storage capabilities.

The goal of Grid Computing is to provide pervasive dependable access to distributed computing resources. The Grid Computing vision, if realized, promises access to computing as easily as people currently access the power grid through their wall sockets. The main focus of SDG is data collection and analysis. But, in order for SDG to work effectively in a joint environment it must also address many of the same issues that face Grid Computing.

Grid computing research focuses on developing an interoperable common infrastructure that provides dependable consistent access to distributed and decentralized computing resources. It addresses the problem of *coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations* (Foster et al., 2001).

To emphasize the focus on developing interoperable tools and interfaces that work across platforms and organizations, Foster (2002) proposed a three-point checklist for grid computing:

- coordinates resources that are not subject to centralized control
- using standard, open, general-purpose protocols and interfaces
- to deliver nontrivial qualities of service.

Below we describe the Globus Toolkit, a open source implementation Grid Computing services (Foster et al., 2002). Through out the paper, we refer back to this description to point ways SDG can potentially leverage these services.

Globus components are classified into 5 types (Foster, 2005):

- *Common Runtime* components provide a set of libraries and tools to allow Globus services to be platform-independent. Many Globus services are based on w*eb services* as defined by the W3 Consortium. These services use XML as the data interchange format, SOAP for messaging and WSDL for service interface description. Some Globus services, like GridFTP that were developed earlier, do not follow the web services framework.
- *Security* components provide services related to user authentication, authorization, secured communications, and credential management. Security is a very important aspect of distributed simulations, but it is not the focus of this paper.
- *Data Management* components provide services related to distributed data management, which includes data transportation, data replication and data access.

- *Information Services* provide registries to allow services to register themselves, to discover other services and to monitor the status of services.
- *Execution Management* components provide the ability to initiate, monitor, manage, schedule, and/or coordinate remote computations. These components can interface with batch job schedulers typically found at supercomputing sites.

Here are the data management components in more detail:

Data movement
- GridFTP provides secure, robust, fast, efficient, standards based data transfer protocol. Version 4 provides striped transfer mode, where multiple nodes works together to transfer their own portion of the file.
- Reliable File Transfer Service (RFT), built on top of GridFTP, is a web service that provides the ability to recover from client-side failure by storing the transfer state in databases. Also, it provides a job scheduler to manage multiple transfers.

Data replication
- Replica Location Service provides a distributed registry that maps *logical file names* to *physical file names*.
- Data Publishing and Replication service provides pull-based services that automatically creates local file replicas based on user request.

Data Access
- Data Access and Integration (DAI) is a federated service that provides 1) registries to discovery data sources, 2) factories to represent data sources, 3) data services to access data source in different formats (relational databases, XML databases, flat files).

DATA REQUIREMENTS AND OPERATIONAL EXPERIENCES FROM URBAN RESOLVE

J9, the Experimentation Directorate of USJFCOM, and the Joint Advanced Warfighting Project, staged several training and integration exercises in early 2004, followed by four experiments, each two weeks long, in June - October, 2004 . Sites participating in the events include:

- TEC Fort Belvoir, Virginia
    - 30 or more workstations
    - Saber - 4 terabytes disk storage
- SPAWAR San Diego, California
    - 20 or more workstations
- J9 DCEE Suffolk, Virginia
    - 50 or more workstations
    - 16 node mini-cluster
- ASC Wright Patterson AFB, Dayton, Ohio
    - GLENN cluster - 128 dual cpu nodes
- MHPCC Maui , Hawaii
    - KOA cluster - 128 dual cpu nodes

The experiments typically ran 5 days a week, 10 hours a day. Simulators might run all night, but with little activity and usually with logging disabled. Depending on availability and requirements, one or both of GLENN and KOA were used. Up to two hundred thousand clutter entities were simulated on the large clusters. Several thousand non-clutter entities were simulated on the other sites. A single node on the large clusters simulated 1000-2000 clutter entities.

Data logging was performed in two modes, near real time and after action. Real time data was inserted in an sqlite database. A node simulating 1000 clutter items would generate an sqlite database of approximately 50 mbytes in an hour. The databases were deleted and reinitialized when they grew to over a gigabyte. If 100 nodes of the cluster were used for clutter simulators, approximately 5 gigabytes per hour of data was generated. For after action use, compressed binary data was stored in an archive directory. Binary compressed data is approximately 7 times smaller than a corresponding database. Each night, the archived data was transferred, via rsync, to Saber, and expanded and decoded into a single MYSQL database.

Clutter data from the GLENN and KOA clusters was not entered into the Saber database, due to size limitations. Data from 100 nodes on GLENN for a 10 day event would have been close to a terabyte. Data from TEC, SPAWAR, J9 and J9 mini-cluster for non-clutter entities were entered into the MYSQL database. Urban Resolve Phase I exercise generated about a terabyte of data in the MYSQL database.

The nightly data transfer was about 15 gigabytes of compressed data. Network transfer rate to Saber was approximately 10 megabits per second. 3 or 4 hours was required to do the rsync transfer. Decoding and indexing the data into the MYSQL database took 12 hours if everything worked perfectly. Human error and other factors usually prevented a days data from being entered into the database before the next days event started. It was usually at least several days after an event before the complete after action database was ready on Saber.

The logging methodology used for the 4 exercises in 2004 was adequate. It was the first attempt at logging data from hundreds of processors distributed geographically around the country simulating thousands of non clutter entities. SDG is intended to remove deficiencies in the 2004 methodology and upgrade what was essentially an experimental system into a production system. The design parameters for SDG specifically address the following list of deficiencies in the 2004 system:

1. Near real time and after action data logging are implemented differently. Near real time queries are restricted by the use of simple aggregators.
2. The use of a single database on Saber does not have the capacity to include clutter data from the GLENN and KOA clusters.
3. Data transfers, decoding and indexing are time consuming and error prone, delaying the availability of the database. A goal is to have the complete database kept up to data continuously.
4. Retrieval of data and database generation for multiple exercises is inconvenient.
5. Expansion to more compute nodes, more entities per compute node and more data per entity is impossible. Disk storage, compute power, and network bandwidth all impose serious limitations.
6. The system does not respond gracefully to hardware and network problems. Saber is a single point of failure which makes all data unavailable.
7. Complex queries which may be useful to analysts are slow or impossible.

Database queries used in Urban Resolve are generally summary in nature. They count how many events or entities (database rows) meet specified criteria. Complex join operations were rarely, or never, used. Were it not for this constraint on the queries, an efficient distributed design would be more difficult.

SDG MODELS

User's Conceptual Model

As described in the previous section, in the current system data collected from the simulation are distributed and replicated at multiple locations. There is one access mechanism to query the data during simulation runtime, and another when the simulation is over. From the user's perspective, these data access complexities are unnecessary.

SDG adds a data access middleware layer that hides these complexities and presents to the user a simple coherent view of data to the user. From their desktops SDG users should be able to access and analyze the data without having to know[1]:

1. How to access the data and what is the network interconnection topology (Access transparency).
2. Where is the data located (Location transparency)
3. Whether the data source has moved (Migration transparency)
4. Whether the data is from a replicated source (replication transparency)
5. Whether data sources are shared (Concurrency transparency).

Users interact with SDG through one of SDG's top-level Managers. Users submit queries to a top-level Manager, and they receive query results from a top-level Manager.

SDG is capable of handling static data sets (no new data added), as well as dynamic data sets (new data continuously being added). For dynamic data sets, users can register canned queries with top-level managers, and receive asynchronous query results.

SDG provides feedback to the users regarding the queries they submit by letting them know:

- Resource usage required to execute the query
- Available resources

Fallacies of Distributed Data computing

In 1991 Peter Deutsch articulated Eight Fallacies of Distributed Computing (Deutsch, 2005):

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. The topology doesn't change
6. There is one administrator
7. The transport cost is zero
8. The network is homogeneous

Distributed software systems developed under these assumptions tend to be brittle. Or, they later have to re-engineered to work around these assumptions.

Potential additional fallacies related to Distributed Data Computing:

1. Disk capacity is infinite
2. Disk latency is zero
3. Disk bandwidth is infinite
4. Data processing cost is zero

1 Tanenbaum & van Steen (2002) defines three additional transparency goals for distributed systems. Some of these goals are beyond the current scope of SDG, and some are not applicable.

The initial design of the 2004 system was based on the assumption that there is sufficient local disk space to store the logged data. But, we found nodes on supercomputer clusters tend to have less local disk space than the average desktop computer. We had to first implement a near-realtime system, then an off-site after action post-processing system.

SDG manages these potential pitfalls through a unique division of labor. SDG hides some of the networking details from the user by explicitly managing the five transparencies listed above, but exposes other details (e.g., resource usage and storage options) to allow users to examine and if needed override default behavior and manage those details themselves.

To address the Distributed Data Computing Fallacies, we intend to provide multiple data services with varying levels capabilities to let the user to select the appropriate services for the tasks at hand. See . These storage options trades-off storage size, query speed and query preprocessing time. For the Urban Resolve exercises approximately 50% of the messages were not logged, because these messages were internal simulation bookkeeping messages. For example, Clutter Intersection (ClutterInt) messages that determine which car should enter the intersection next usually are of no interest to the analysts.

Typically, a user may want to include a compressed storage option to keep an archive of the simulation data. Then, the user may wish to select another storage option, for all or partial subset of the messages, for faster querying. If the user chose multiple storage options, he has the option of deleting/truncating a data storage to recover disk space.

Designer's Conceptual Model

SDG Managers perform all of the data access/ query/management tasks. Conceptually, there are three types of Managers: top-level, data source and worker.
Top-level Managers have published addresses. Users connect through the top-level managers. To minimize network traffic typically there is at least one top-level manager for each local area network. Top-level managers know how to connect to each other. Non-top-level managers know how to connect to at least one top-level manager.

Data Source managers store the actual data. Other applications to insert data into Data Source Managers through defined APIs.

Worker managers perform most of the work within the system. When given a data processing task, the top-level manager decomposes a task into sub-tasks

- Enlists one or more managers. These managers can be worker, data source or other top-level managers.
- Assigns sub-tasks to these managers and data source managers
- Defines a data flow topology linking together the sub-task executions.

The mapping of the tasks onto managers must take into account and take advantage of a heterogeneous computing environment. The networking infrastructure within a local cluster typically uses Gigabit Ethernet, or even faster proprietary Myrinet. The inter-cluster networking infrastructure is typically orders of magnitude slower. Computation must be moved closer to the data sources to avoid transportation penalties.
Also, the storage hierarchy varies from cluster to cluster. For example, the original configuration of the Koa cluster at Maui Supercomputing Center does not include local hard drives. A 10TB Storage Area network (SAN) mount on GFS functions as the only secondary storage. The Glenn cluster at ASC has a total of 10TB storage mounted on local hard drives, and 10TB SAN storage mounted on GFS.

USE CASES

In this section we describe various use cases to capture the functional requirements for SDG. Also, we describe how to map these requirements to Grid Computing functionality provided by GLOBUS. We divide the use cases into three categories:

1. SDG system administrators: how to manage itself
2. Data administrators: how to manage the data sets
3. Data analysts: how to query/analyze data

Storage Options	*Storage Size*	*Query Speed*	*Query pre-processing*
Do not log	Zero	N/A	None
Compressed (raw)	Small	Very slow	Small
Text (decoded)	Medium	Slow	Small
Database	Medium	Medium	Medium
DB w/ indexing	Large	Fast	Large
Cube (D = # of dimensions)	Large, for high D	Fast	Large, for high D

Table 1 Range of storage options that trades-off storage size, query speed and preprocessing time.

System Administrators

Startup/shutdown SDG Nodes. Remotely manage and control SDG managers.
Monitor SDG Node status. Remotely monitor the health of SDG managers.

These system administration functionalities match well with Globus' Execution Management components and Information Services' Monitoring and Discovery System (MDS). MDS's Index Service is able to register services, as well as maintain resource properties associated with the service. MDS's Trigger Service can be used to send alerts when certain conditions occurs, such as when a local disk is nearing capacity. Execution Management provides ways to submit, cancel and manage remote job executions.

Data Administrators

Monitor Data/Resource Usage Statistics
- Monitor the rate and size of data flowing into SDG
- Monitor the available disk capacity
- Monitor network usage
- Monitor CPU usage

Archive data sources. Copy/move data sources into one centralized location. This is useful for archiving data into a centralized SAN or a tape archival system.

Merge/split data sources.
- Combine multiple data sources into one source.
- Partition one data source into multiple data sources.

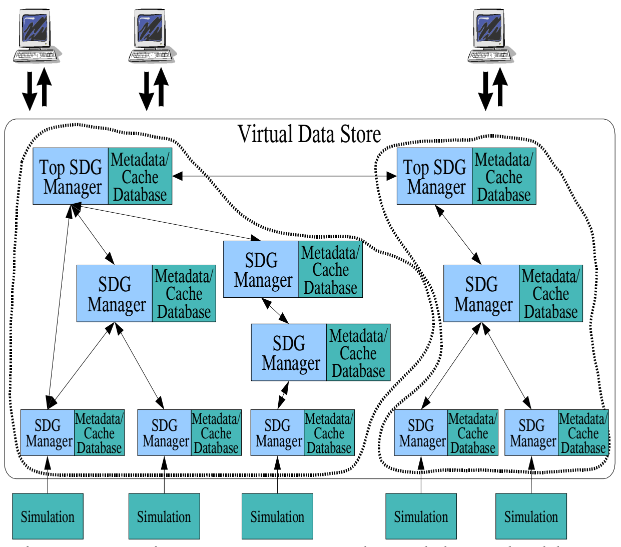


Figure 1 Designer's Conceptual Model. Dashed loops indicates the boundary of local area networks.

These operations are useful to take advantage of parallelism when extra compute resources are available.

Here we again plan to leverage MDS. MDS is able to interface with cluster monitoring tools, such as Ganglia (Ganglia, 2005), to produce up-to-date system load/usage information. In addition, GridFTP, Reliable File Transfer service and Replica Location Service also play important roles.

Data Analysts

Select Queries. User sends a *select* SQL query to a top-level manager. The top-level manager returns the query result in a result set table. The underlying data is stored in multiple locations, but to the user it appears to be one big centralized database.

We further classify select queries into:
- Aggregate: operators like sum, min, max, average
- Union: more than one table and/or sub-query
- Simple: does not involve aggregates or unions

For simple select queries SDG managers only need to concatenate results, returned by sub-tasks, without further processing.

Sample simple queries include:
- Return all entity weapon damage reports within the last 30 min.
- Return red tank movements within the last 10 min.

Sample union queries include:
- Return all entities that were painted by a sensor
- Return marking information of the entity that fired a weapon within the last 10 minutes

Sample aggregate queries include:
- Count the sensor tracks grouped by sensor type, or group by degree of assuredness
- Return killer/victim scoreboards
- Return sensor/target scoreboards

Resource Usage Explanation Queries. This is similar to MySQL's EXPLAIN command. Given a select query, SDG traces through the execution of the query, and explains which resources and how much were used to answer the query.

Canned Queries. User defines periodic/trigger select queries. Based on the defined period, or the trigger, SDG executes the query and asynchronously returns the result to the user.

Sample uses:

- Receive alerts when a missile is launched
- Automatically update killer-victim scoreboard when there is a weapon fired

The GLOBUS' Data Access and Integration (DAI) service provides a common web services interface for accessing heterogeneous data sources (files, relational, xml). The relational part of the interface allows clients to submit SQL queries to Data services and to receive query result sets from the Data services. Also, DAI supports asynchronous delivery, which may be useful for periodic canned queries. In addition, Distributed Query Processing (DQP) service layers distributed join capabilities on top of DAI.

However, one key reservation we have about using DAI/DQP is the overhead of using SOAP/XML based communication for query result processing. Using standards based communication make sense if the data sources are heterogeneous. But, in our case we are focused just on relational data. Furthermore, scalability to handle very large simulation data sets is one of our overriding concerns.

DISTRIBUTED MULTIDIMENSIONAL ANALYSES

In this section we focus the implementation of select aggregate queries, such as the Sensor/Target Scoreboards.

Background: Sensor/Target Scoreboards

One of the key focus of Urban Resolve Phase I is to study the effectiveness of future ISR sensors in helping soldiers operate in complex urban environments. The Sensor/Target (S/T) Scoreboard provide a visual way of quickly comparing the relative effectiveness of individual sensor platforms and sensor modes against different types of target. As we shall see S/T Scoreboard is a specific instance of the more general multidimensional analysis.

In the Urban Resolve federation, a simulated sensor entity lays down sensor footprints to delimit sensor coverage sweep. For each target entity within the footprint, a *contact report* is generated to hold the result of the sensor detection. The contact report includes information about the sensor entity, the platform the sensor entity is mounted on, the sensor mode, the target entity, the detection status, the perceived target type, the perceived target location, the perceived target velocity and so on.

Sensor/Target scoreboards have the capability of providing summary views by aggregating individual sensor platforms into sensor platform types, such as *high altitude*, *medium altitude*, and *low altitude*. And, it aggregates individual target entity objects into target classes, which can range from the generic (like *Civilian Large Trucks*) to the specific (like Russian *MAZ-543 MEL*). As described in (Graebener, 2003), the current implementation of the scoreboard provides four levels of details. The information provided are:

1. Table of contact report counts broken down by sensor platform types and by target classes.
2. Given a sensor platform type and a target class, table of number of contact report counts broken down by sensor platforms and by sensor mode.
3. Given a sensor platform and a sensor mode, list of target objects.
4. Given a target object, list detailed target object attributes.

Initially, the S/T Scoreboard displays the level one aggregate table of sensor platform types and target classes. By clicking on a table cell (i.e., specifying a particular platform type and target class), the S/T scoreboard brings up the level two display of sensor platforms and sensor modes. Sensor modes are methods detection, such as Moving Target Indicator (MTI) and Synthetic Aperture radar (SAR) Spot and SAR strip.

Analysis of S/T Scoreboard from Multidimensional Perspective

The current implementation of S/T Scoreboards projects the contact reports along three dimensions for analysis. The three dimensions are sensor platform, target object and sensor mode. In addition, sensor platforms are aggregated into sensor platform types, and target objects are aggregated into target classes. As Figure 2 graphically depicts these three dimensions as linear partial orderings. These dimensions can be crossed to create lattices, as show in Figure 3.

With respect to the right lattice in Figure 3, the information contained in level one S/T scoreboard correspond to the node *tc*, sensor platform type by target class. The level two information correspond to slices of node *pcm*, where *p* is restricted to a particular sensor type *t* and entity class *c* is given. Levels three and four correspond to target objects specified by cells in node *pom*.

The four levels of the S/T scoreboard presents information useful to the analysts. But, other nodes within the lattices maybe of potential interests. For example, node *cm* summarizes the effectiveness of sensor modes with respect to target class. Or, node *pc* summarizes the effectiveness of sensor platforms with respect to target class. In addition, other dimensions not used in S/T Scoreboards may be of potential interest, for example detection status, time, location, terrain classification (high-rises, low-rises, flat), weather condition, and so on.

Multidimensional Analysis

S/T Scoreboard falls into a analysis classed called multidimensional analysis, or sometimes called On-Line Analytical Processing (OLAP) or data warehousing (Kimbal *et. al.*, 1998). Other types of scoreboards, like Killer/Victim and Truth/Perception, are also multidimensional in nature. Conceptually, the data structure used to store multidimensional analysis data is the *cube*. The 2D array data structure used to store a 2-dimensional scoreboard is extend to n dimensions.

Users could query the OLAP system for the entire cube, but typical the users are more interest in projections and partial views of the cube. Operations on the cube include:

- roll-up: aggregate data along a dimension to hide details. This correspond to walking up the dimension lattice.
- drill-down: partition data along a dimension to reveal more details. This correspond to walking down the dimension lattice.
- Slice & dice: select subsets of the cube elements.

Query and Data Characteristics

Query and data characteristics within the simulation differ from traditional OLAP assumptions in two significant ways:

- Query is concurrent with insertions
- Data is distributed

Typically, OLAP is performed on historical data. For example, retail chains may keep sales transaction records to determine their best performing stores, or emerging consumer trends. This analysis is usually performed off-line. The analysis need not be updated as individual sales transactions occur.

In addition, data is typically sent to a centralized facility to be analyzed. In our case, it is not feasible to centralize the data because of the amount of data and near-realtime nature of the query. Our data is logged locally at the point of generation. If there are 100 simulation nodes, then we have 100 local logs.

Previous works have studied distributed OLAP implementations (Goil and Choudhary, 2001; Beynon *et. al.*, 2002). Typically, they employ some type of data partitioning scheme to perform load balancing and/or to reduce I/O overhead. For example, in the *chunking* data partitioning scheme the data cube is partitioned into smaller sub-cubes. The number of dimensions of the sub-cubes remains the same, but now each dimension holds just a subset of the possible dimension values.

However, in our case these data partitioning schemes are not applicable. Our data partitioning is dictated by the simulation. Moving these messages creates network traffic that may disrupte the actual simulation. Since we are not able to preposition data, we are investigating cube compression techniques to minimize storage and the I/O needed to aggregate the local cubes. These techniques include partial cube materialization (Harinarayan, 1996) that selectively pre-computes a subset of lattice nodes; and coalesced cubes (Sismanis and Roussopoulos, 2004; Sismanis et al., 2002) and shell fragments (Xiaolei et al., 2004) that offer compact ways of storing the cube.

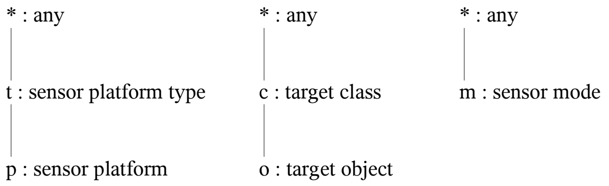


Figure 2 Three possible dimensions to partition the data for analysis.

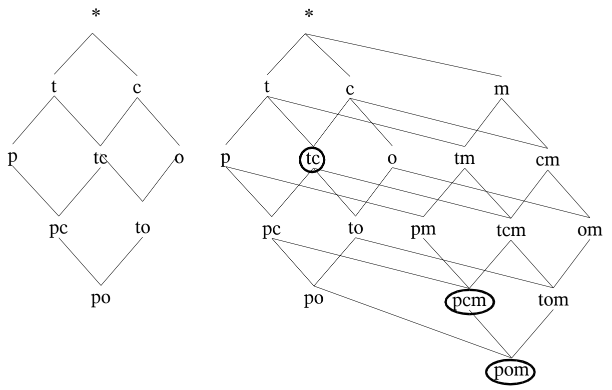


Figure 3 Lattices are generated by crossing the dimensions. Crossing the sensor dimension with the target dimension generates the lattice on the left. Crossing the left lattice with the sensor mode dimension generates the lattice on the right.

IMPLEMENTATION STATUS AND EXPERIMENTAL RESULTS

Implementation of a Simple Distributed Sensor Target Scoreboard

OLAP systems on single processors are widely used and described in the literature. Two implementations are frequently used. MOLAP stores multidimensional data in an explicit multidimensional structure. ROLAP stores multidimensional data in a relational data base. MOLAP provides faster access to data. ROLAP stores sparse data more efficiently. We chose ROLAP for the implementation of SDG.

We develop the system and implement features in an incremental fashion in order to deliver capabilities to J9 in a reasonable fashion. This has the added benefit of providing feedback which can be applied to future development. We identified the sensor target scoreboard, discussed earlier, as a critical feature representative of many key features that would ultimately be required, and that could be implemented quickly and efficiently.

We chose one week of archived data from one Urban Resolve event as test data. The sensor target scoreboard is prepared from a table in the database named I_ContactReport. We are interested in deriving a unique value for 1) the type of sensor, 2) the type of target and 3) the detection status, for each row of the table. This information is used to create a 3 dimensional table of counts for each unique combination. Other information is discarded at this time. Future enhancements will incorporate information such as time and location to create a 5 (or larger) dimensional table.

The sensor type is identified by 8 columns in the contact report: platform_type_EntityKind, platform_type_Domain, platform_type_CountryCode, platform_type_Category, platform_type_Subcategory, platform_type_Specific, and platform_type_Extra, sensor_mode.

Similarly, target type is identified by 7 columns in the contact report: detected_type_EntityKind, detected_type_Domain, detected_type_CountryCode, detected_type_Category, detected_type_Subcategory, detected_type_specific, and detected_type_Extra.

The detection status is identified by a single column named detection status.

For testing, we recreated a database from one week of a 2004 event, The I_ContactReport table has approximately 18 million records. One column, node, identifies the machine on which the row was generated. To simulate the distributed generation of the data we created 4 new databases based on applying a regular expression to the value in the node column. Only the 16 columns listed above were copied to the 4 new databases. 2 additional columns, detected_enum and sensor_enum were added to identify unique combinations of the target and of the sensor.

Next, a procedure was applied to each of the 4 new databases. In a real exercise, this procedure would be applied independently and concurrently on each computer maintaining a database.

The procedure consists of the following steps using MYSQL commands:

1. create a table of unique (distinct in MYSQL terminology) combinations of sensor values and unique combinations of target values. Assign an enum to each.
2. Create a row in the table for each combination of sensor type, target type and detection status that occurs in the I_ContactReport table. Compute a column count giving the number of times the combination occurs. With appropriate indexing this takes 6 minutes for for 6 million records in one of the 4 sub-databases.
3. Add rows to the table for “wildcards” as appropriate for a data cube. A row is created for any sensor, any target, any detection status, three wildcards. This should equal the number of rows in the contact table for a 3 dimensional table. Do the same for all permissible use of two wildcards and one wildcard.

This procedures is applied when a new dataset is introduced to the system. It is then applied to any new data which is added to the system. The data cube is always up to date.

There are now 4 relatively small databases containing complete and nearly instantaneously accessible information on the count of any combination of sensor types, target types and detection types. A user query to a top level data manager is relayed to low level data managers connected to each of the 4 subdatabases. The responses are merged by combining responses with the same dimension value and summing the count field. The result is returned to the user.

CONCLUSION

The use of large clusters (hundreds nodes or more) of processors to meet the demand for high performance computing is becoming a mature technology. The extension to use multiple clusters is likewise common, but less mature. The use of OLAP technology to analyse large data sets also is becoming a mature technology. To support USFCOM and JAWP we have combined distributed clusters and OLAP. Using this approach, and innovation in key areas, we are able to support our customer's current and expanding needs. A key principle is to store data close to it's source to minimize network traffic. A second principle is to utilize the computational and storage resources of distributed clusters for database functions. These two principles reinforce, rather than interfere with, each other in the design and implementation of the data grid. A key area of innovation is the intelligent Manager. The intelligent Manager categorizes a query, creates an execution plan, distributes the work for the query, aggregates the results, and delivers the results. The queries required and commonly used by JSAF analysts are efficiently executed by this system. Fault tolerance and realistic data archiving are additional benefits of our implementation. We will maintain and extend the system to include more processors, more clusters, larger datasets and more robust queries as required by our customer.

ACKNOWLEDGEMENTS

The authors wish to acknowledge the members of the ISI JESPP team who have contributed to this paper through their efforts and their intellectual stimulation. Much of the success reported here came from the Joint Experimentation on Scalable Parallel Processor project, initiated, directed and funded by the Joint Forces Command and to a very large degree conducted on the compute assets of the Maui High performance Computing Center, ASC-MSRC at Wright Patterson Air Force Base and other members of the High Performance Computing Modernization Program.

This material is based on research sponsored by the Air Force Research Laboratory under agreement numbers F30602-02-C-0213 and FA8750-05-2-0204. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes, notwithstanding any copyright notation thereon.

REFERENCES

Baralis, E., Paraboschi, S., & Teniente, E. (1997). Materialized View Selection in a Multidimensional Database. VLDB Conference, 1997.

Barbara, D., Sullivan, M. (1998). Quasi-Cubes: A space-efficient way to support approximate multidimensional databases, Technical report, ISE Dept., George Mason University.

Beynon, M., Chang, C., Catalyurek, U., Kurc, T., Sussman, A., Andrade, H., Ferreira, R., Saltz, J. (2002). Processing large-scale multi-dimensional data in parallel and distributed environments, *Parallel Computing.*

Deutsch, P. (2005). The Eight Fallacies of Distributed Computing. http://today.java.net/jag/Fallacies.html

Foster, I., Kesselman. C., & Tuecke, S. (2001). The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications, Vol. 15, No. 3*, p200-222.

Foster, I. (2002). What is the Grid? A Three Point Checklist. *Grid Today*, vol. 1, no 6. http://www.gridtoday.com/02/0722/100136.html

Foster, I., Kesselman, C., Nick, J., & Tuecke, S. (2002) The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. *Open Grid Service Infrastructure WG, Global Grid Forum*, June 22, 2002.

Foster, I. (2005). A Globus Toolkit Primer. http://www.globus.org/toolkit/docs/4.0/key/GT4_Primer_0.6.pdf

Ganglia (2005). http://ganglia.sourceforge.net/

Graebener, R.J., Rafuse, G., Miller, R. & Yao, K.T. (2003). Successful Joint Experimentation Starts at the Data Collection Trail. I/ITSEC 2003.

Graebener, R.J., Rafuse, G., Miller, R. & Yao, K.T. (2004). Successful Joint Experimentation Starts at the Data Collection Trail—Part II. I/ITSEC 2004.

Goil, S., & Choudhary, A. (2001). PARSIMONY: An Infrastructure for Parallel Multidimensional Analysis and Data Mining, *Journal of Parallel and Distributed Computing*, v61.

Harinaryan, V., Rajaraman, A. & Ullman J.D. (1996). Implementing Data Cubes Efficiently. Proc. ACM SIGMOD '96, p 205-216.

Pedersen, T. B., & Jensen, C. S. (2001) Multi-dimensional Database Technology. IEEE , Dec 2001.

Riedewald, M., Divyakant, A., El Abbadi, A. (2001). Flexible Data Cubes for Online Aggregation. *Proceedings of the 8th International Conference on Database Theory*.

Tenenbaum, A. S. & van Steen, M. (2002). *Distributed Systems principles and paradigms*, New Jersey: Prentice Hall.

Appendix C

High Performance Computing Facilities for Joint Military Simulation Data Management

Dan M. Davis
Information Sciences Institute, USC
Marina del Rey, California
ddavis@isi.edu

Garth D. Baer
George Washington University
Washington, District of Columbia
garthbaer@gmail.com

ABSTRACT

The overwhelming amount of output data inundating many in the simulation user community is a widespread problem. Much of this torrent is generated by current high-end computational capabilities. Joint and combined forces analysts are faced with the major tasks of first validating and then utilizing the data generated by modern techniques. A major part of the solution is an optimized data management software architecture. To enable the analysts to achieve success commensurate with the users' goals, a dedicated and appropriately designed data management facility was required. Taking cognizance of the advances made in the physical sciences' community, such a facility was conceived, designed and is being proposed to the HPCMP. The techniques of identifying, quantifying and implementing important data-handling parameters should be applicable to many large data-set problems in the Test and Evaluation community.

This paper will discuss the general state-of-the-art in data management, the specific problems presented by the U.S. Joint Forces Command simulations of up to a million independent SAF entities on a global-scale terrain, the methods used defining the problems presented thereby, and the path to the decision to standup a new facility. Adopting the successful techniques found effective in basic science, *e.g.* studying approaches used by other scientific research efforts, effective data management schemes have been discovered. Both the design process and the architecture itself will be laid out. Some issues addressed will be the choice of compute platform, the provision of associated communications, the selection of storage peripherals, the analysis of incipient technical advances that are likely to be germane, cost-benefit analyses of competing installations and the approach necessary in order to design for the future. Specific performance, cost and operational issues will be presented and analyzed. Lessons learned from this evolution should be extensible into many fields associated with modeling and simulation, as well as the T&E community in general.

ABOUT THE AUTHORS

Dan M. Davis is the Director, JESPP Project, Information Sciences Institute, University of Southern California, and has been active for more than a decade in large-scale distributed simulations for the DoD. While he was the Assistant Director, Center for Advanced Computing Research, California Institute of Technology, he managed Synthetic Forces Express. He has also served as a Director at the Maui High Performance Computing Center. He served in the USMC on active duty and is a Commander, Cryptologic Specialty, U.S.N.R.-Ret. He has been the Chairman of the Coalition of Academic Supercomputing Centers and received a B.A. and a J.D., University of Colorado, Boulder.

Garth D. Baer is a technical analyst who is studying the impact of policy on technology as well as the changes technology makes on policy formulation and implementation. He currently is a graduate student at The George Washington University, studying policy formation at the Federal level. Previously a Principle Support Engineer at Oracle Software Corporation, he developed new web-based applications and troubleshoots production database issues. Earlier, he was a Mission Control Engineer for Milstar Communications Satellites at Lockheed-Martin. He participated in a multi-university group developing a vision statement for DoD policy on M&S. He received Bachelors in Physics, Univ. of Colorado, Boulder and a Masters in Technology Management, Colorado Technical Univ.

BACKGROUND

Since before the advent of written history, military commanders have sought ways to prepare their fighters for upcoming battles. Some of this involved working out plans of attack and some involved assessing the various levels of capability. Though today's commanders use different tools, their goals would be recognized by the commanders of yore. Both would like to observe their fighters in something akin to real combat and make judgments based on those observations. But the advent of industrial power has added to their burdens and our current commanders have had their job skills extended into something more akin to logisticians than their ancient counterparts. The terrible swiftness of modern swords allows little room for contemplation and adjustment.

To help ameliorate this problem, the US Joint Forces Command (JFCOM) has been designated as the transformation laboratory of the US Armed Forces. They have adopted the well-tested, yet powerful and easily modified, Semi-Automatic Forces (SAF) family as one of their major simulation platforms. The version they principally use is Joint SAF, or JSAF. One of the research objectives of their Joint Experimentation Directorate (J9) is to assess the capabilities of various systems when deployed in an urban setting, they thus must "field" an urban population in experiments such as Urban Resolve (UR). This is a complex problem, calling into use all of the capable skills of the simulation programmers, system designers, experiment operators, and data analysts.

Typically, a single IA-32 based PC (like Intel Pentiums) running on a Linux operating system can support a few thousand civilian clutter entities and lashing 20 to 30 of these computers on a Ethernet Local Area Network (LAN) can support a population of around 30K (Ceranowicz, 2002). Looking at any major urban center in the world, one would see that each has nearly two orders of magnitude more entities in action than that, *e.g.* Baghdad has a population of 7.8M with vehicles adding at least another million. In the vernacular of the computational scientists, the LAN solution does not scale to this level. New hardware and software is needed. The JFCOM Joint Experimentation on Scalable Parallel Processors (JESPP) Project was initiated to respond to this problem (Lucas, 2003). Utilizing the readily available capabilities of the High Performance Computing Modernization Program, (HPCMP) the JESPP team was successful in achieving the needed scalability on Linux clusters, also using IA-32 architecture.

Thence came the deluge. Like others in the SAF community, JFCOM analysts now faced increasingly large data sets, complex sensor results, convoluted scenarios, and diverse environments. The need to re-run certain actions of interest also heightened the need for data collection, processing, management, storage and retrieval. Literally terabytes of information could be anticipated, and even that assumed much of the specific clutter background activity was not archived and therefore could not be exactly duplicated. This is true because the underlying SAF codes are not deterministic, relying on pseudo random numbers to determine a action's result. Though this deluge of data increases each simulation's reliability, managing it proved a difficult task.

The JFCOM leadership sought the assistance of the Information Sciences Institute of the University of Southern California (ISI/USC) and the Center for Advanced Computing Research of the California Institute of Technology (CACR/Caltech) to resolve these data problems. As reported elsewhere in this conference, (Bunn, 2005) their approach is based upon their experience with High Energy Physics sensor data.

In most high performance computing implementations, the data analysis is performed on platforms designed to facilitate the creation of the data or to interface easily with sensors and other data-producing devices. The JFCOM situation is somewhat different. They bear the pressures of several calls upon their services. These include:

- analysis of battlefields of the 2015 time-frame
- designs intended for use in the next year
- real-time assistance to the warfighter today

These will likely also obviate the use of their clusters for analysis.

That brings about a new opportunity for JFCOM and HPCMPO: not only using custom designed database software, but developing an optimized set of Linux cluster nodes to facilitate the analysts' task and ensure the validity and sanctity of the data. The rest of this paper is directed at establishing the needs of the users, surveying the potential platforms for this use, assessing the various performance characteristics and designing a stable and efficacious system.

THE PROBLEM

Data

The authors do not intend to burden the reader with a surfeit of arcane technical descriptions of the format of the data produced by JSAF and associated sensor federates (see Graebener, 2003 for more details.) That which follows is a higher-level and, the authors contend, more germane general description of the data available for collection and the steps necessary to render it useful to the analysts and experiment controllers. This should allow the reader to better assess similarities between their simulations and the simulation under study. This is intended to provide a basis for analyzing the relevancy of this work to theirs.

For simplicity's sake, we break down the JFCOM simulations into four broad areas:

- Terrain and Environment
- Civilian "clutter"
- Operational entities
- Intelligence sensors

Of these, only the first broad category, Terrain and Environment does not represent a major data task. This is due to the fact that the terrain is a largely static entity and the environmental variables are often easily duplicated without storing the actual values during the experiment, *e.g.* the day/night interface is easily recovered, while the impact of clouds may be more random and need to be recorded.

On the other hand, the amount of data presented by the clutter of civilians can be huge. Positions of pedestrian entities and vehicle models are reported to the system on the order of once every 10 to 100 milliseconds. The system itself can tolerate up to 500 millisecond latencies before showing strain. In any case, being able to simulate millions of entities, (Barrett, 2004) now presents the problem of what to do with all of this location data (in three dimensions), orientation data (in three axes) and state data (color, type, damaged, dead, …) that can report as often as 100 times in a second. The JESPP team has developed a powerful algorithm for saving data so generated (Wagenbreth, 2005) but the volume of data is so high that one very real solution is to simply discard the "clutter" data, treating it much like environmental data.

Next are Operational Entities. These are largely armed forces of US, Allied and enemy units. JSAF, like all of the SAFs, presents the possibility of very good Human-In-The-Loop (HITL) intervention. That means that JFCOM military personnel can personally control US and Allied forces and a "Red" team can bring to the experiment all of the creativity and experience they have garnered, frequently after decades of experience in the service. This brings to the surface a significant difference in JFCOM data and, say Bank of America "transaction data" or Caltech "physics sensor data." The solutions sought in both the software and hardware areas must be sufficiently general to support any conceivable data type and load, yet sufficiently specific to optimize both areas for JSAF use. This is not a trivial conundrum.

Finally, there is the data to be gleaned from the intelligence sensors simulation programs. This issue is exacerbated by Intellectual Property issues, occasioned by the interest the programmers have in the intelligence sensor program, which is proprietary. The nature, amount and relevancy of this data is to some very real degree outside the control of the either the authors or the other managers engaged on this project.

The System

One of the great strengths of the JFCOM experimental design is its distributed and dispersed nature. The experiments themselves are housed and controlled by the JFCOM out of its experimental bay near Suffolk, Virginia. Environments and data are managed remotely out of Fort Belvoir in Northern Virginia. The civilian clutter are laid down and managed by a team a continent away in San Diego, at the SPAWAR center on Point Loma. The two 128 node, 256 processor Linux clusters that are provided by the HPCMP, are located in Maui at the Maui High Performance Computing Center (MHPCC) and at Wright Patterson Air Force Base at the Aeronautical Systems Center Major Shared Resource Center (ASC-MSRC) in Ohio.

Communications between the sites are provided by the Defense Research and Engineering Network (DREN). Experiments for Urban Resolve have been unclassified, but work for CENTCOM may require encryption of the communications' links. The Linux operating system is common across the net (usually Fedora) and most of the programs are written in C++, with a smattering of Java. While the sites at Suffolk and San Diego are entirely devoted to J9, the two HPCMP sites are multi-tasked, including maintaining the dedicated J9 cluster, named *koa* and *glenn*.

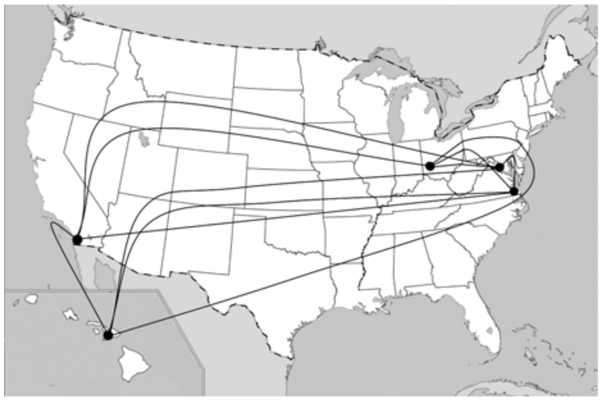


Figure 1. Notional Multi Path WAN between, TEC JFCOM, ASC-MSRC, SPAWAR, and MHPCC

Note that there are geographical dispersion issues, Maui being on the order of five thousand miles from Suffolk, as indicated in the notional diagram in Figure 1. This precludes easily and economically meeting with the entire staff. Further, with a five time zone span, the operational synchronization is difficult, most especially in the summer when the mainland sites go to daylight savings time, while Hawai'i does not, thereby creating a six-hour difference.

The Users

The users are nearly as diverse as the system is dispersed. The description that follows will lay down some of these differences. All of these users are potential generators and users of the data to be managed:

- System administrators
- Simulation operators
- Friendly and Red Team controllers
- Analysts

The system administrators must effective manage the distributed hardware and voluminous software necessary to maintain the simulation. As load balancing and fault tolerance are not automatic, they must have real-time access to data during the experiments. After the experiment, they must have unfettered access to the stored data to make assessments of necessary upgrades and remedial repairs. One consistently taxing area for the last decade has been the issue of configuration control across as many as a dozen supercomputers, *e.g.* ensuring the compiler version is in synch.

The simulation operators themselves are tasked with delivering the requisite Forces Modeling and Simulation (FMS) capability to the experiment managers and scenario designers. For this, they need to be able to understand what areas of the simulation are functioning and which areas are producing anomalous data. In addition to this real-time data, they need after-action review to allow reprogramming JSAF's and other federates' code to either correct errors or to enhance existing or add new, capabilities.

The next group, the controllers, need to have carefully limited access to data. They should be given only that data which serves the goal of providing them that which they would have on a real battlefield. Most often, this information will be presented to the controllers in the form of appropriately formatted "messages," emulating real communications.

Finally, we have the analysts, for whom most of this work has been done in the first place. These analysts have varying needs, varying time requirements and varying local compute capabilities. During the experiment, they want to have real-time access to all of the data in the experiment. They may want to discover the exact location, orientation and state of any entity being simulated. They may want to be presented with the output of fairly sophisticated data queries, (Graebener, 2004). They may want to view the real-time Plan View Display (Map) (Figure 2) or Stealth (3-D) to see how the experiment is going. The analysts may order the entire action to be repeated, from the beginning or from some critical point.

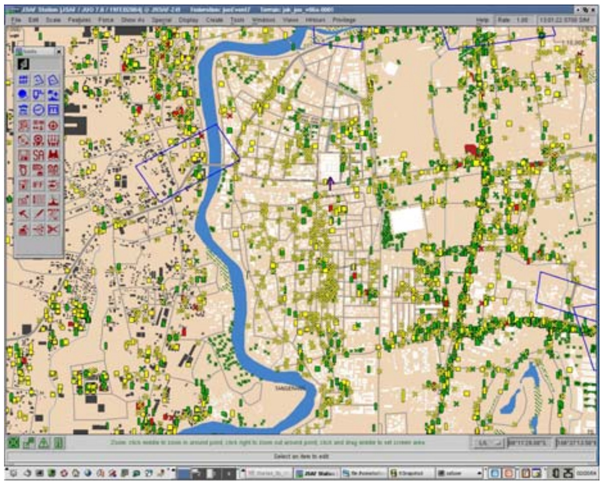


Figure 2. Plan View Display from JSAF

This, however, is only the first stage of the analysts' needs and goals. Every night, during operations stand-downs, the analyst may be reviewing the previous days results with a eye toward altering the experiment or re-running important parts.

After the action is done, the analysts may spend weeks reviewing the data. They will use SQL commands to elicit as much information as possible from the data collected. Again, they may want to repopulate the scenario and run some parts again. They may find entirely new areas of interest or study. They may want to compare this experiment with ones run a year or more before. They may also wish to embark upon extensive and unconstrained data mining, by its very definition, a process that has no preconceived goal, therefor no clear view of the type, nature or extent of the data that will yield up new and vital insights. (Davis, 2004)

Implementation

The JESPP team has developed code that is capable of intercepting, decoding, archiving and organizing all of the data generated (Yao, 2005), and that system is currently under the process of being parallelized to make it more scalable. With the user set distributed across the country (Virginia to Hawai'i), the system is being developed to respond equally to many users and many analysts accessing the data simultaneously.

The system is designed with MySQL as the database engine and Linux as an operating system. Both are open source programs, commonly available and well documented and supported. The use of one of the popular Unix operating systems ensures easy portability to other high performance computing platforms, such as Cray, IBM, SGI, HP and others.

While design decisions have not been finalized, it seems likely that some of the MeshRouter communications structure, so successful on the simulations itself, will bring fault tolerance, scalability, and supportability to the distributed user-base. Another open issue along this line is whether to have the major data facility located at one of the existing sites, a new site or distributed amongst all of the current sites. (Gottschalk, 2005)

OPTIONS

High Level Architecture

In looking at these issues, one of the first considerations is that of the general system architecture being considered. One possibility would be to have the entire system designed for the single processor PC used by the analyst. Putting this burden on the analysts' PCs would severely constrain the potential value of the simulation.

Another option would be to utilize some of the nodes of the distributed compute (DC) facility at MHPCC and ASC-MSRC. As discussed before, this would limit operational capabilities and make the administration of *koa* and *glenn* even more difficult. It would, however, have the benefit of being readily available and, for that reason, it may be the interim solution.

The final concept is the establishment of a new analytical computational facility and the development of a scalable program. This would allow the system designers to parallelize, not only separate and independent inquiries, but also to distribute the difficult and time-consuming functions of the data-collection and data-analysis utilities. The location of such a facility is in question. As the issues involved here may be comparable to the reader's??, a modest effort will be made to lay them out. Because some of the data may be classified, at least one classified facility is indicated. Due to the fact that the users may actually come from anywhere in the world, having mirror sites that are widely dispersed may be useful. On the other hand, there arguably is a significant benefit to having the facility under one roof and within easy access of the JFCOM staff.

Compute Platform

Recent developments in CPU architecture may impact the choice of the compute platform. Both AMD and Intel have fielded CPUs capable of 64 bit memory addressing, the Opteron and Pentium 64, respectively. As an aside, it should be noted the previous generation Athlons and Pentiums used 64 bit registers, but 32 bit memory addressing. Both companies still produce pure 32 bit implementations, as well as hybrid implementations, the Hyper-Threaded (HT) configuration. While the 64 bit addressing in large database implementations may seem attractive, current offerings do not allow the user to take much advantage of the larger address space.

Looking at these solutions *seriatim,* the vast majority of FMS compute platforms today are based on the advanced design chip designs of the IA-32 architecture. Some of these, such as the Xeon chips from IBM, are capable of breaking the "4GB" memory addressing space of typical IA-32 by a margin (up to 64 GB) that is not reflected in available board support, usually restricted by the limited number of RAM slots and board chipsets to 8 GB total. The 64 bit Itaniums (often regarded as an architecture that will not last,) Opteron, Xeons, and Athlon 64s are on the market and available, but again, in ISI's experience, appear on boards limited to 8GB. Memory bus speeds are up and memory prices are down, allowing economical configurations on the PC model that are then distributed across nodes on Linux clusters.

There are other architectures in the offering. The DARPA HPCS program is currently pursuing three advanced designs that will feature shared memory in the PetaScale machines. (Graham, 2004) To adequately use these, there are programs underway to prepare the systems and programming software to support the new architectures. (Kepner, 2003) Not only will these advances reduce latencies and increase bandwidth in data storage, processing and retrieval, the new architectures may revolutionize the code development paradigm now in vogue. The parallel programmer, faced with distributed processing and with distributed memory has long since had to develop very convoluted and sophisticated designs to enable programming for systems that are often distributed across thousands of compute nodes, in dozens of machines, located all across the United States and spanning half a dozen time zones. (Brunnett, 1998.) Many have argued that these advanced architectures would, to some degree, obviate these tortuous paths to useable code.

Database Software

As this article focuses on database installation choices and analysis, only a few major parameters will be discussed here, vis-à-vis the selection of the database software. The first high-level choice is the option to go with one of the major database companies, with their elaborate contractual obligations and cost, but with very professional support staffs and considerable high quality documentation and training. Some of these commercial packages are quite powerful and capable, while others are clearly intended for home use only and have neither the power or the scope to handle serious database burdens, as are found in FMS. The authors have had the experience of working on projects that erroneously began using these "lower-power" databases, only to quickly have to abandon them in favor of something more capable. This, of course, led to much lost training and programming time.

The second possibility is the utilization of the plethora of really substantial and fully capable open source and public license software. Two of these are MySQL and SQL Lite. As with much of the software associated with the Linux revolution, there is an active and vocal (at least electronically) community, supporting and discussing these choices and their uses. Programmer support is easily obtained and new users have, within the author's experience, developed proficiency and sophisticated approaches rapidly.

Quantification of Platform Performance

While it is unlikely to evoke enthusiastic responses and effusive thanks, the prudent user will be well advised to set aside enough time and personnel to rigorously do early testing to ascertain the acceptability of the performance of the target platform. Following the lead of Paul Messina of Caltech, (Messina,1990) the authors accept the tenet that performance testing is always most likely to provide reliable data if the benchmark used is not an artificially generated "toy " program. The most successful performance testing will be the use of the software to be used, processing the data to be studied, in operational conditions as close to real as possible.

Table 1. Computers Evaluated

Computer	**Athlon1800**	**Xeon**	**Opteron64**
CPU Spd.	1.5GHz	3.02 GHz	2.4 GHz
CPUs/Node	2	2	2
L2 Cache	128	256	256
RAM	2 GB	2 GB	2 GB
Bus Sp	100	133	133
HDD	60 GB	60 GB	240 GB
RPM	7200	7200	7200

For assessing the proposed facility for JFCOM use, the authors had normalization runs conducted on three basic platforms, an Intel Pentium III without multi-threading, a dual processor Intel Xeon with multi-threading and an dual Opteron 64, also exhibiting multi-threading. Other parameters and specifications are given in Table 1. While the runs represented here do not seem adequate to the authors, they are indicative of the types of testing that would be appropriate.

The test scenarios were of a standard analysis of the processing of data from a JSAF run, accomplished for JFCOM. It deals with location, orientation, status and movement of "clutter", that is civilian or non-military pedestrians and vehicles (Ceronowicz, 2002). Varying conditions were input into the data processing evolution and the following results were obtained. The data management program makes several decisions about whether to convert (decode) the raw data and store it as more easily retrieved ASCII text, to throw it away or to archive it without decoding. A common position is to discard any data for which insufficient processing power is available. This Hobson's choice is unacceptable in the eyes of the authors as presently discarded data may hold key information yet unbeknown to the user (Davis, 2004).

The chart in Figure 3 shows the number of Decoder Delete events that were processed in 15 minutes.

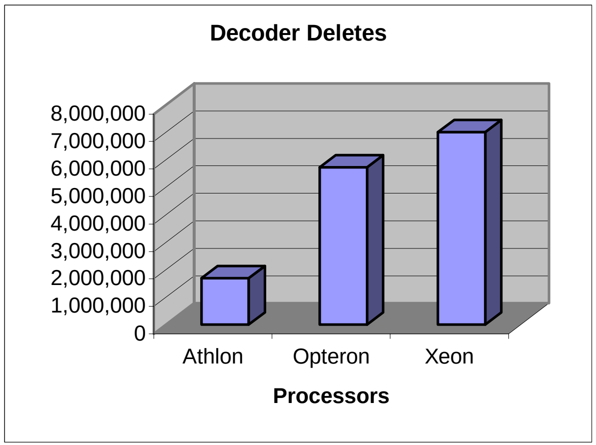


Figure 3. Decoder Deletes – 15 Minutes

This activity consisted of scanning through populated tables looking for the oldest data and leaving everything else in the table. As can be seen the older and slower Athlons are significantly out-performed by the newer, 64 bit oriented processors. However, it should be noted that the Athlon dual boards differed in other ways as well. Perhaps the major insight from this slide is that the H/W configurations do make a marked difference in performance and even simple initial tests can begin to show performance break points.

Next, attention is turned to the truncating of the tables. Often, during JFCOM simulations, when the data tables became full or when data was no longer needed, the truncate command was issued, thereby deleting all data in the table. This tasked another processor consumed with a housekeeping task of consequence in this particular analytical process. Figure 4 shows the truncating results, totaled, for a ten minute run.

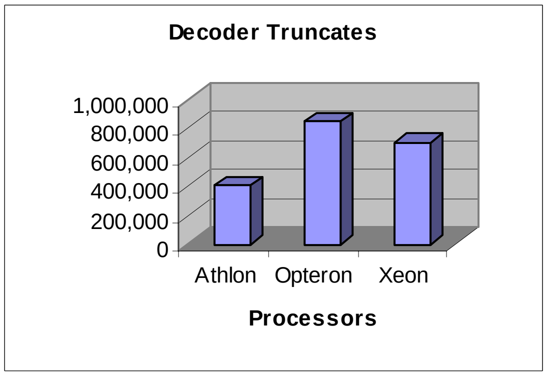


Figure 4. Decoder Truncates – 10 Minutes

Side by side comparisons for any smaller units of time suffered due to the inconsistent rate of the activity over time. One graph is presented here to show the "spikey" quality of the data processing loads. Figure 5 is a graph of a single run of the Athlon machine doing decoder truncates. It demonstrates the necessity of accumulating performance over several minutes time.

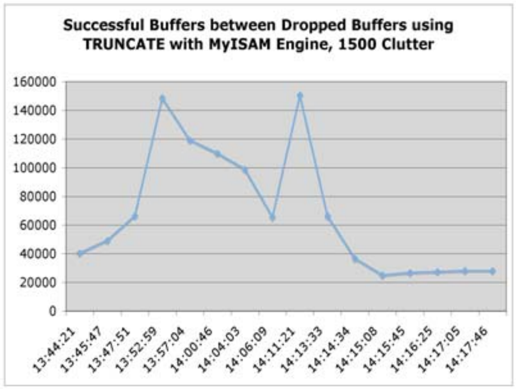


Figure 5. "Spikes" of Athlon doing Truncates

The fact that the application used here as an exemplar of the approach did or did not produce dramatically different performance levels should not be taken as a de-motivating factor in the desirability of performing such tests. This result is supported by a survey of database users who clearly favored software advances. Often, dramatic differences have been observed and it is critical to avoid a trap of procuring millions of dollars in Linux cluster hardware only to find the 64 bit option was either an additional expense with no benefit or, on the other hand, an expense that would have paid for itself in the first few months of operation. Again, the main message here is that benchmarking should be done with the target application, not a general benchmark tool.

DISTRIBUTED DATA PROCESSING

The next topic to discuss is the design characteristics of the JFCOM usage that militated in favor of a distributed data facility. (Yao, 2005) A quick review of the JFCOM operational paradigm may be well advised here. JFCOM runs large scale simulations that are supported by and of interest to personnel at facilities that are literally world-wide, but are typically distributed from the Peninsula of Virginia to the slopes of Haleakela in Maui. This use is dynamic in both its load and its subject matter. During simulations, data is produced at vastly different rates and data is accessed and analyzed in similarly unpredictable ways. Original, non-scalable, designs for data handling were structured around discarding data of too great a volume to be managed, then returning data assumed to be of interest to a data facility at JFCOM in Suffolk and processing it there. This virtually precluded data analysis and use during the operations period of the simulation experiment. Figure 6 below is a notional representation of the operations phase of the experiment, with the data facility's (SABER's) grey tint indicating its idle state.

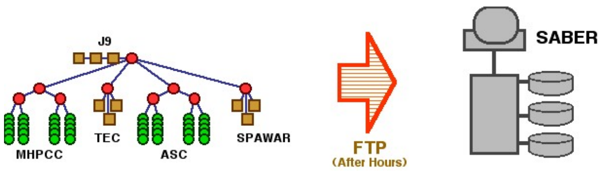


Figure 6. Current JFCOM Data Design - Ops

The converse becomes true during the data management cycle of the experiment, typically during the night and on weekends and then the entire period of analysis between the experiments. As seen in Figure 7 below, now the operations platforms are idle (at least as far as JFCOM is concerned) and the data facility is fully utilized.

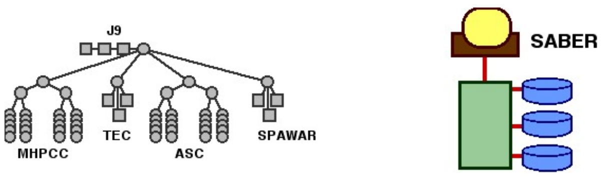


Figure 7. Current JFCOM Data Design - Analysis

Some other weaknesses of this system may be obvious, but merit attention. Firstly, a single data location is not fault tolerant for the diverse and dispersed user community, as equipment failures anywhere between the user analyst and the data results in total downtime. Secondly, this concentrated asset does not lend itself to scaling, as intervening tree architectures put huge loads on the "root" node, making it a likely (and observed) choke point (Barrett, 2004). Thirdly, the current facility processes everything serially at that point, not taking advantage of the benefits afforded by parallel processing Linux Clusters.

One of the design characteristics being studied at this time is the possibility and desirability of using:

- dispersed Linux clusters
- partitions of the operational cluster
- second processors on cluster nodes
- other variants

These would allow assessment of the optimal dispersion of the data and data handling.

The concept of using dispersed supercomputers is often called met-computing, a close relative of the currently popular term grid computing (Foster, 1997). The meta-computing configurations most favored by the JFCOM team is one in which the major compute sites, Maui and Ohio, would be likely locations for much of the data logging, decoding, storing, organizing, and archiving, thereby minimizing wide area network (WAN) communications. This could be significantly augmented by a separate analytical Linux cluster at either a new location, at one of the compute sites or at JFCOM in Virginia. Figure 8 is a notional view of the operations phase of such a design. Note that the user (or any number of users) can access the data during the simulation experiment. Using the scalable communications hierarchy developed for operational scalability, (Gottschalk, 2005), the number of users accessing the three data facilities should not be impeded by communications bottlenecks.

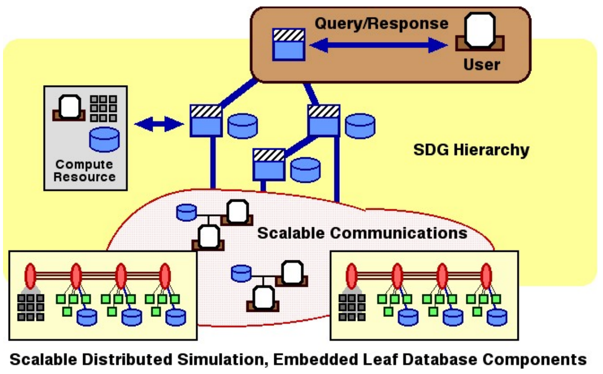


Figure 8. Planned JFCOM Data Design – Ops

Further, as additional Linux cluster nodes have been set aside or new cluster nodes provided at each site, the data logging, decoding and entry into the MySQL database will occur real time, allowing for near real time accesses by analysts. This is, of course, a huge improvement, allowing the analyst to retrieve and internalize insights from on-going operations with an eye toward real time changes in the experiments operation.

As in real live-fire actions, the amount of information available to the soldier and commander alike is growing far beyond their ability to make optimal use of it. This is also reflected in the simulation environment. Originally, subject matter experts reviewed, discussed and opined about the validity of and insights from the simulations. Now, the data is just too vast and the action too complex to make best use of this type of analysis.

Both SME and the analysts have a need for all the machine processing assistance that can be provided to them. This, again, will help model future assistance to the warfighter in combat. It should be remembered that machine processing and analysis are important when more than a million entities may be simulated across terrain databases that are literally global, but may also be engaged in high fidelity urban simulations, such as the snap shot in Figure 9.

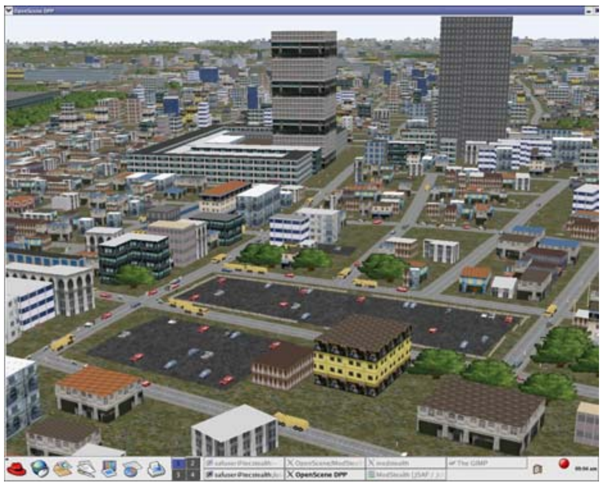


Figure 9. JFCOM Experiment – One Block of City

The last configuration diagram shows the usage during the analytical phase of the work. As can be seen in Figure 10, there is a significant amount of activity, with only the simulation operational nodes being idle. In actuality, these nodes are turned back over to the HPCMP community for other batch or interactive users. This is represented here in by the absence of the nodes in the two lower boxes, which represent the compute sites. Of course, as analytical capabilities are implemented, even these "operational" nodes may be pressed into service.

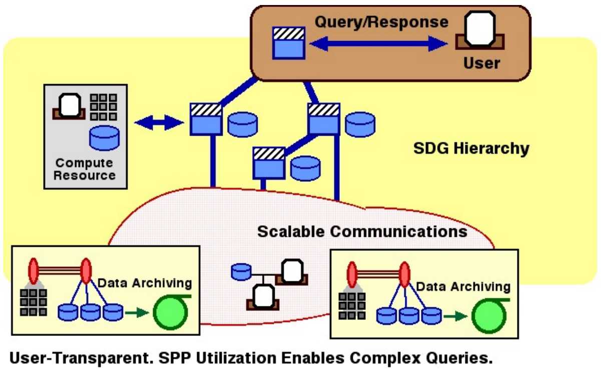


Figure 10. Planned JFCOM Data Design –- Analysis

It is not suggested that the efficacy of this design for JFCOM makes this design appropriate for all situations, but the process described above will serve the simulation professional well. Careful planning begins with establishing and maintaining good communications with all of the experimenters and all of the potential users. Repeated "white board" planning sessions and pilot trials are very useful in determining the correct configuration. Fortunately, one of the real benefits of the Linux cluster revolution is that its component parts are nearly universally useful, so even design decisions that later prove less than optimal can be recovered, due to the ease with which the cluster nodes can be put to alternative, beneficial uses.

CONCLUSIONS

The authors started from what they consider to be a virtually unassailable set of premises:

- Modeling and simulation can utilize the benefits from high performance computing
- This benefit produces so much data that an unaided user will find it difficult to extract, process and analyze the results
- Users and computing facilities are often dispersed geographically
- Failure to recover all of the possible insights will likely be an undesirable result for the analytical teams and the warfighters

The authors' beginning thesis could be stated as follows: the curse of the surfeit of data can best be met by the very creator of that curse, *i.e.* scalable, dispersed, parallel meta-computing. Pursuant to that thesis, the JESPP team has created and initially tested a distributed data-handling platform that is both scalable and responsive.

Part of that design process is the scoping, evaluating and implementing the facilities themselves. While ideally these issues are completely outside the ken of the user, they are of significant concern to the simulation team and the computational scientists supporting them.

Lessons Learned

The first lesson learned is the desirability of seeking advice, counsel and support from as diverse a group as possible. Resisting the temptation to do it *de novo*, it has proven to be very useful to seek as broad an input as could be arranged. The benefits of the University community came to the fore here, as major research Universities have on their staffs experts in every field.

A second lesson, flowing from the first, was that the experience and insights from the High Energy Physics community were more germane than were the commercial transaction-processing programs or the Internet recreational data-search designs. The reason for this is assumed to be the closer relation to the types of data, the technical literacy of the users, the more uniform access to high-bandwidth, and the lack of the need for elaborate inter-user security, but very high external security. However, there are differences from the HEP community, *e.g.* the simulation community's broader and more dynamic analytical interest. The HEP community tends to work with output from a single, well-defined experiment, looking for a specific set of data. The military Forces Modeling and Simulation community has many user interests and changing goals.

The third lesson is that working with open-source, public licensed software has many advantages for the developer. The Linux community is active and involved. Source code is available for scrutiny, modification, and implementation. Hundreds of hours that would be expended in the procurement process are now available for more productive endeavors. While the authors recognize the value added by major commercial vendors and their support staffs, the JFCOM experience indicates that open source software should be seriously considered.

The last lesson learned to be offered here is the desirability of simultaneously pursuing both a "bottoms-up" and "top-down" approach. The exigencies of operational necessity dictate that current but evolving code bases, be pursued and the authors' experience indicates that these efforts often outstrip and outperform new "improved" designs. This bottoms-up approach keeps the simulation going and serves to be a real time laboratory to continuously assess the applicability of new concepts. The top-down approach is useful in supplying a more theoretically founded view of the ultimate design. This approach keeps the design from growing without focus, necessary infrastructure or control. This is also very important in insuring scalability of the final product. Pending arrival of new computer architectures, parallel processing offers the best hope of increasing compute power. If these implementations do not scale, however, all of the additional processors will be of little use.

In closing, it should be restated that High Performance Computing brings FMS new capabilities, which bring new floods of information, for which HPC facilities can be designed. The combination of careful planning and openness to others skills are a *sin qua non* of success.

ACKNOWLEDGEMENTS

The authors wish to acknowledge the members of the ISI JESPP team who have contributed to this paper through their efforts and their intellectual stimulation. Much of the success reported here came from the Joint Experimentation on Scalable Parallel Processor project, initiated, directed and funded by the Joint Forces Command and to a very large degree conducted on the compute assets of the Maui High performance Computing Center, ASC-MSRC at Wright Patterson Air Force Base and other members of the High Performance Computing Modernization Program. Dr. Thomas D. Gottschalk and Dr. Ke-Thia Yao were very generous in sharing their designs for distributed data management. Craig Ward of ISI generated the performance characterization data. Without the support and encouragement from all of the above, none of this would have been possible. This material is based on research sponsored by the Air Force Research Laboratory under agreements numbers F30602-02-C-0213 and FA8750-05-2-0204. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes, notwithstanding any copyright notation thereon

REFERENCES

Barrett, B. & Gottschalk, T., (2004) Advanced Message Routing for Scalable Distributed Simulations . *Proceedings of the Interservice / Industry Training, Simulation and Education Conference*, Orlando, FL.

Brunett, S., Davis, D., Gottschalk, T., & Messina, P. (1998) Implementing Distributed Synthetic Forces Simulations in Metacomputing Environments, *The Seventh Heterogeneous Computing Workshop*, Orlando, FL.

Brunn, J. & Gottschalk, T. (2005, publication pending), Incorporating High Energy Physics Data Capabilities into Joint Forces Simulations . *Proceedings of the Interservice / Industry Training, Simulation and Education Conference,* Orlando, FL.

Ceranowicz, A., Torpey, M., Hellfinstine, W., Evans, J. & Hines, J. (2002) Reflections on Building the Joint Experimental Federation. *Proceedings of the Interservice/Industry Training, Simulation and Education Conference,* Orlando, FL.

Foster, I. & Kesselman C. (1997) Globus: A Metacomputing Infrastructure Toolkit. *Intl J. Supercomputer Applications*, 11(2), 115 –128.

Gottschalk, T. & Amburn, P., (2005, publication pending), Extending The MeshRouter Framework for Distributed Simulations. *Proceedings of the Interservice / Industry Training, Simulation and Education Conference*, Orlando, FL.

Graham, S., Snir, M., and Patterson, C., Editors, (2004) *Getting up to Speed, the Future of Supercomputing,* The National Academy Press, Washington, D.C.

Graebener, R. & Rafuse, G., Miller, R., & Yao, K-T., (2003) The Road to Successful Joint Experimentation Starts at the Data Collection Trail. *Proceedings of the Interservice / Industry Training, Simulation and Education Conference,* Orlando, FL.

Graebener, R. & Rafuse, G., Miller, R., & Yao, K-T., (2004) The Road to Successful Joint Experimentation Starts at the Data Collection Trail, Part II. *Proceedings of the Interservice / Industry Training, Simulation and Education Conference,* Orlando, FL.

Kepner, J., (2003), HPC Productivity: An Overarching View, *International Journal of High Performance Computing Applications,* London, UK

Lucas, R. & Davis, D. (2004) Joint Experimentation in Scalable Parallel Processors, *Proceedings of the Interservice / Industry Training, Simulation and Education Conference*, Orlando, FL.

Messina, P., Baillie, C. F., Felten, E. W., Hipes, P. G., Walker, D. W., Williams, R. D., et al**,** (1990), Benchmarking Advanced Architecture Computers, *Concurrency,* 2, 195.

Yao, K-T. & Wagenbreth, G., (2005, publication pending), Simulation Data Grid: Joint Experimentation Data Management and Analysis, *Proceedings of the Interservice / Industry Training, Simulation and Education Conference*, Orlando, FL.

Wagenbreth, G., Davis, D., Gottschalk, T., Lucas, R. & Yao, K-T. (2005, publication pending), Operational Experience, Distributed Simulations, Data Management and Analysis , *Proceedings of the 2005 Winter Simulation Conference*, Orlando, FL.

Appendix D

Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

ENABLING 1,000,000-ENTITY SIMULATIONS ON DISTRIBUTED LINUX CLUSTERS

Gene Wagenbreth
Ke-Thia Yao
Dan M. Davis
Robert F. Lucas

Information Sciences Institute,
4676 Admiralty Way
University of Southern California
Marina del Rey, CA 90292, U.S.A.

Thomas D. Gottschalk

Center for Advanced Computing Research
1200 E. California Boulevard
California Institute of Technology
Pasadena, CA, 91125, U.S.A.

ABSTRACT

The Information Sciences Institute and Caltech are enabling USJFCOM and the Institute for Defense Analyses to conduct entity-level simulation experiments using hundreds of distributed computer nodes on Linux Clusters as a vehicle for simulating millions of JSAF entities. Included below is the experience with the design and implementation of the code that increased scalability, thereby enabling two orders of magnitude growth and the effective use of DoD high-end computers. A typical JSAF experiment generates several terabytes of logged data, which is queried in near-real-time and for months afterward. The amount of logged data and the desired database query performance mandated the redesign of the original logger system's monolithic database, making it distributed and incorporating several advanced concepts. System procedures and practices were established to reliably execute the global-scale simulations, effectively operate the distributed computers, efficiently process and store terabytes of data, and provide straightforward access to the data by analysts.

1 INTRODUCTION

As background, the authors set out a brief review of the entity-level simulations used by the Joint Forces Command (JFCOM) and others in the Department of Defense. The need for increased scope and resolution is covered, which led to the consideration of high performance computing (HPC). They will then cover the architectural constraints, the implemented designs, and performance testing and analysis. This will set the stage for the major thrust of the paper: the experiences with operations and data management. The paper concludes with a look at future work and suggests some conclusions that may be drawn at this time.

The United States is facing an entirely new environment of politics, economics, threats and conflict settings. (Barnett, 2004). This is forcing a deep and broad re-evaluation of its defensive posture. In looking to the future, the DoD has a vested interest in being able to simulate more than one million vehicles, each with sophisticated, independent behaviors. This is driven by the government's needs to effectively use new computer and communications technology in the U.S. defense organizations (Cebrowski, 1998) and simulate more complex human functions (Ceranowicz, 2004) in technically diverse situations (Sanne, 1999). The U.S. Department of Defense (DoD) has begun a series of experiments to model and simulate the complexities of urban environments. In support of their mission, analysts need to conduct interactive experiments with entity-level simulations, using programs such as the Semi-Automated Forces (SAF) family used for years by the DoD (Ceranowicz, 2002).

This needs to be done at a scale and level of resolution adequate for modeling the complexities of military operations in urban areas. All of this leads to the analysts' requirement of simulations of at least 1,000,000 vehicles or entities on a global-scale terrain with high-resolution insets. Experimenters using large numbers of Linux PCs distributed across a local area network (LAN) found that net communications constraints limited the analysts to tens of thousands of vehicles, about two orders of magnitude fewer vehicles than their needs. This paper addresses the benefits of the successful application of computational science and parallel computing on High End Computers to this situation. By extension, it lays out a template for those with similar simulation needs, who can make beneficial use of such computers, one group of which is often called Scalable Parallel Processors (SPPs). JFCOM used assets of the High Performance Modernization Program (HPCMP.)

While there are many Forces Modeling and Simulation (FMS) approaches that are currently in use, experimentation at the entity level provides some very attractive features, both for training and for analysis. Making these simulations so that the user can directly involve humans, *i.e.* Human-in-the-Loop (HITL), additionally augments the DoD's ability to assess true impacts on personnel and procedures. (Ben-Ari, 1998) There are several new methods to modeling human behavior (Hill, 2000). While these require significant independent research (vanLent, 1998), they also require significant additional computing power. Current PC capability does not allow the analyst to conduct these experiments at the scale and level of resolution necessary and much of the FMS community currently reports not using High Performance Computing (HPC). (Davis, 2004) These constraints and disinclinations have also been found in other varieties of simulation. (Kaufman, 1993)

In the present case, JFCOM's newfound emphasis on civilian entities, called "clutter," has extended the horizons of entity-count requirements by approximately two orders of magnitude. In any urban setting, the number of civilian vehicles will easily outnumber the combat vehicles by a factor of ten, and more likely, by a factor of 100. Trying to assess the utility of sensors and the efficacy of intelligence analysis in discriminating the combatants from the civilians will putatively be insufficiently served by a simulation that is limited to a few thousand vehicles total.

The current work on the authors' Joint Experimentation on Scalable Parallel Processors (JESPP) Linux clusters enabled the successful simulation of the required 1,000,000 entities. Software implementations stressing efficient internode communications were necessary to achieve the desired scalability. Further, this is a challenge that the authors assert may only be amenable to meta-computing across widely dispersed and heterogeneous parallel computer assets (Foster, 1997). One major advance was the design of both the "Tree" and the "Mesh" software routers to efficiently route information between simulators on local and wide area networks.

The work reported on in this paper is based on the earlier work funded by DARPA in the mid nineties, headed by Paul Messina at Caltech (Messina, 1997). The Synthetic Forces Express project (SF Express) was conceived and initiated to explore the utility of SPPs as a solution to the communications bottlenecks that were then being experienced by one of the conventional SAFs, ModSAF. The SF Express charter was to demonstrate the capability of practically hosting a scalable communications architecture on multiple SPPs to simulate 50K vehicles: an order-of-magnitude increase over the size of an earlier major simulation, Synthetic Theater of War–Europe (STOW-E).

The professionals from the SF Express effort, many of whom are on the JESPP staff, were able to mount a simulation run, with more than 100,000 individually simulated vehicles in March of 1998. The runs used several different types of SPPs at nine separate sites spanning seven time zones. These sites were linked by a variety of wide-area networks. (Brunett, 1998)

That work was built on the DIS standard utilized by the SAFs at that time. That standard was replaced by the HLA/RTI standard that was purportedly more scalable, but several years of use has shown the clear limits of this new approach. This has not prevented some experimenters from getting very good results while simulating approximately 30,000 entities (Ceranowicz, 2002). These new standards and additional requirements have driven the development of the two new router designs, Mesh and Tree Routers.

1.1 JSAF

The Joint SemiAutomated Forces simulation suite is used by the US Joint Forces command in its experimentation efforts. JSAF runs on a network of processors, which communicate via a local or wide area network. Communication is implemented with High Level Architecture (HLA) and a custom version of Run-Time Infrastructure (RTI) software version RTI-s. A run is implemented as a federation of simulators or clients. Multiple clients in addition to JSAF are typically included in a simulation. The typical user-interface is a map visualization or a three-dimensional, rendered view as are shown in Figure 1.

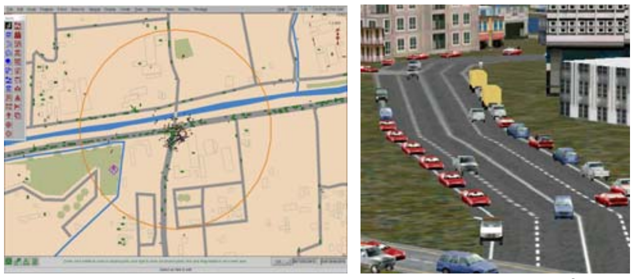


Figure 1: Plan View and 3D Rendered Displays from a SAF

HLA and RTI use the publish/subscribe model for communication between processors. Typically, these processors are in relatively powerful PCs using the Linux operating system. A data item is associated with an interest set. Each JSAF instance dynamically subscribes to ranges of interest. A JSAF may be interested in, for example, a geographic area or a range of radio frequencies. When a data item is published, the RTI must send it to all interested clients.

A typical JSAF run simulated a few thousand entities using a few workstations on a LAN. A simple broadcast of all data to all nodes is sufficient for this size simulation. The RTI on each node discarded data that was not of interest to each receiving node. Broadcast is not sufficient when the simulation is extended to tens of thousands of entities and scores of workstations. UDP multicast was implemented to replace the simple broadcast. Each simulator received only the data to which it has subscribed, *i.e.* in which it has a stated interest.

Operational imperatives drive experimental designs that now require further expansions of JSAF capabilities such as more entities, more complexity, larger geographic area, multiple resolution terrain, and more complex environments. The most readily available source of increased compute power, up to one or more orders of magnitude, is the capability presented by Scalable Parallel Processors. In the JESPP project, JSAF was implemented on multiple Linux clusters, using hundreds of processors on each cluster. Future runs will require thousands of processors on multiple clusters. The obstacle to using these resources effectively was the poor scaling of the original inter-node communication.

Both Tree and Mesh software routers were implemented on individual nodes within the local mesh network that also included all of the client simulators. Each simulator is connected to only one router. Routers are connected to multiple clients and multiple routers. Two types of information are present: data along with interest description and the current interest state of each client. The interest state changes as each node subscribes and un-subscribes to specific interest sets.

In this architecture, each router must maintain the interest set of each node to which it is connected, including other routers. A router's interest set is the union of all connected nodes. A router then uses the interest state associated with data it receives to determine how to forward the data. For a given topology, communication is minimized such that each client node receives exactly the data in which it is interested.

The initial router implemented in JESPP was a tree router. Each router has multiple clients but only one parent. There is one router that is the top of the tree. A second topology has subsequently been implemented that the authors refer to as a mesh router, which had shown scalability across 1900 nodes in SF Express (Brunett, 1998). Instead of a single router at the top of a tree, there is a mesh of routers with "all-to-all" communication. Each simulator is a client of one of the mesh routers. Like the tree router, the primary task of the mesh router is to maintain the interest state of all clients, forwarding only data that is of interest to each client. Further hybrid topologies should be possible with little or no code modification.

The ultimate goal is to establish the capacity of a simulation to scale easily as the number of processors is increased by several orders of magnitude. Comprehensive testing and measurement is required to document the performance of various topologies and router implementations. This testing identifies performance bottlenecks and suggests alternative implementations to be tested. Multiple simulation scenarios are required to construct guidelines for assigning simulators, routers and topologies to multiple SPPs. The simulations and entities had to be designed such that nodes subscribe mainly to a local subset of information.

JFCOM's Experimentation Directorate's recent Joint Urban Operations (JUO) experiments have demonstrated the viability of Forces Modeling and Simulation in a distributed environment. JSAF and the RTI-s communications system, provides the ability to run distributed simulations with sites located from Norfolk, Virginia to Maui, Hawai`i. Interest-aware routers are essential for communications in these distributed environments. The current RTI-s framework provides such routers connected in a straightforward tree topology, which is successful for small to medium sized simulations, but faces a number of significant limitations for large simulations over wide area networks. In particular, traffic is forced through a single site, drastically increasing distances messages must travel to sites not near the top of the tree. Aggregate bandwidth is limited to the bandwidth of the site hosting the top router, and failures in the upper levels of the router tree can result in widespread communications losses throughout the system.

To resolve these issues, this work extends the RTI-s software router infrastructure to accommodate more sophisticated, general router topologies, including both the existing tree framework and a new generalization of the fully connected mesh topologies used in SF Express. The new software router objects incorporate the scalable features of the SF Express design, while optionally using low-level RTI-s objects to perform actual site-to-site communications. General communications protocols are incorporated in the Mesh Router architecture in such a way that it has no impact on the application software. The (substantial) limitations of the original mesh router formalism have been eliminated, allowing fully dynamic operations. The mesh topology capabilities allow aggregate bandwidth and site-to-site latencies to match actual network performance.

1.2 Large Scale Forces Modeling and Simulation

Recent experiments within the Joint Forces Command's Experimentation Directorate, J9, demonstrate the feasibility of forces modeling and simulation applications in a large field of play with fine-grained resolution. As mentioned previously, simulating such battle spaces requires large computational resources, often distributed across multiple sites. The ongoing Joint Urban Operations (JUO) experiment utilizes the JSAF application suite and the RTI-s Run Time Infrastructure to scale to over 300 federates distributed across the United States (Ceranowicz, 2002). The JUO exercise has shown the scalability of the JSAF/RTI-s infrastructure and of interest-based, router-managed communication. At the same time, the simulation has highlighted a need for improvements in the communication architecture.

The current JUO network topology is a tree of software routers (see Figure 2 for wide area network diagram). The hub and spoke network model introduced by this tree infrastructure increases latency between distributed sites and exposes the entire network to a single point of failure. The tree topology also poses a scalability limitation within the distributed sites. It is the authors' belief that an improved routing infrastructure is required for the continued success of large-scale entity level simulations, particularly as entity counts as well as complexity and fidelity increase.

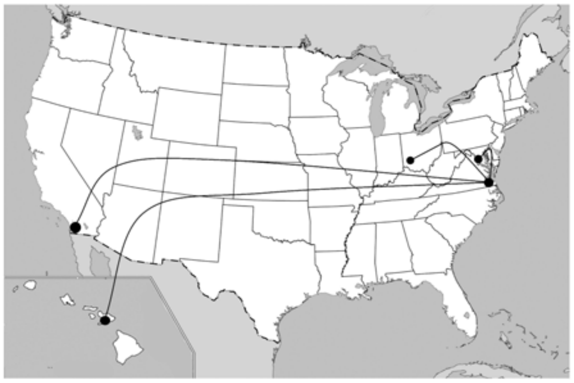


Figure 2: Software Routing Topology for the JUO Exercise

1.3 Scalable Parallel Processors

The JUO exercise requires a computational ability unavailable using traditional groups of workstations. SPPs provide the required computational power, with modest increase in development and execution effort (Lucas, 2003). Common SPPs include the IBM SP, SGI Origin, Cray T3E, and the "Beowulf" Linux clusters. Traditionally, SPPs provide services not available in a group of workstations: high speed networks, massive disk arrays shared across the entire resource, and large per-CPU physical memory. In addition, SPPs generally have uniform environments across the entire machine and tools for scheduling, management and scalable interactive control (starting processes across 100 nodes should take the same amount of time as it does across 10).

Linux clusters have recently become suitable platforms for the high performance computing community and are, therefore, readily available at Department of Defense HPCMP Centers. These clusters are ideal platforms for use in the JUO exercise because of their close heritage to the Linux workstations used in the interactive test bays. Although there is additional software to tie the cluster into one SPP, the basic libraries, compiler, and kernel are often the same on a cluster as on a workstation.

1.4 RTI-s

RTI-s provides the HLA Run Time Infrastructure (RTI) for the JUO federation. RTI-s was originally designed in hopes it would overcome the scalability and performance limitations found in RTI implementations at the time and even greater limitations found in the Distributed Interactive Simulation (DIS, IEEE 1278). RTI-s is arguably not a fully compliant HLA/RTI implementation, a matter of less and less consequence these days. Specifically, it does not implement timestamp ordered *receives*, ownership transfer, and Management Object Model (MOM) interactions. In addition, federates discover new objects at first update, rather than at creation time. The JSAF applications are receive-ordered by design and are optimized to respond best to delayed object discovery, so these limitations are not constraining in the existing environment.

Point-to-point modalities in RTI-s use distinctly separate routing processes for communication. The routers provide data distribution and interest management for the federation, which would be too heavy for a simulator to handle. Presently, the tree topology (Figure 3) is used for connecting routers. A tree presents a simple structure for preventing message loops, as there are no potential loops in the system.

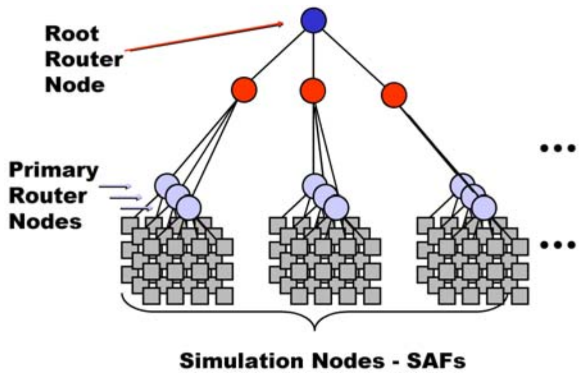


Figure 3: Tree Topology Used by RTI-s for Point-to-Point Message traffic

The Synthetic Forces Express (SF Express) (Brunett & Gottschalk, 1997) project first demonstrated the suitability of both the SPP and mesh router concepts for discrete entity modeling. The SF Express project extended the ModSAF simulation engine (Calder, 1993), focusing on the communication protocols to extend scalability.

At the SC'96 conference in November 1996, the SF Express team achieved a 10,000-vehicle simulation using a single 1,024-node Intel Paragon machine. Message routing within the SPP used the Message Passing Interface (MPI) (MPI Forum, 1993). Later work allowed the code to run on multiple SPP installations across a variety of networks by introducing gateways between SPPs. The gateway routers were connected using UDP. With these improvements, the project was then able to field a simulation of 50,000 vehicles using 1,904 processors over six SPPs.

The structure of the SF Express router network is shown in Figure 4 These routers distribute (PopUp) and collect (PullDown) messages from client simulators outside the Primary's client set. The SF Express architecture scales to increased problem size by replicating the basic triad and adding full up/down communication links among the triads, as shown in Figure 4. This architecture is the progenitor of the JESPP architecture.

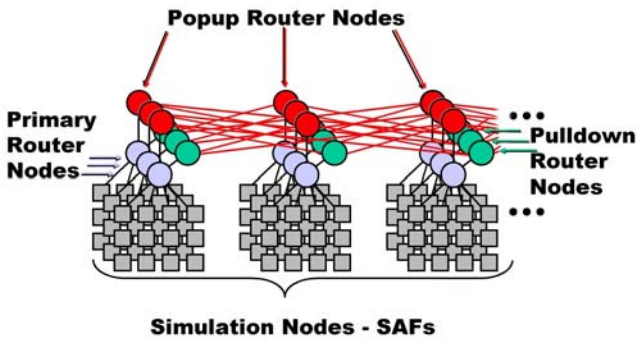


Figure 4: Basic Building Blocks of the Scalable Design Used in the SF Express Routing Network

As a final *magnum opus* accomplishment, the Caltech/ISI/JPL team was able to run a 108,000-entity simulation in March of 1998. These simulated entities were full ModSAF Operational vehicles: tanks, trucks, Bradleys, HumVees, etc. This was demonstrated at a DARPA meeting and was stable enough to be scheduled to be run when it was needed. It used 13 different heterogeneous parallel computers, located from Maryland to Maui, spreading across six time zones.

While the SF Express project was dramatically successful, it had no life beyond a number of 50K-100K entity simulation demonstrations. One issue was the need of one live operator to effectively control each 20 tanks. While the tanks behaviors were correct within local operational areas, they did not have sufficient behavioral sophistication to plan and execute long-range missions. Many of these issues were anticipated, but for a number of reasons they were not seen as invalidating the project, and they did not detract from the proof of concept offered by the success in demonstrating scalability.

This approach seems to have been validated, as new missions have now mandated the provision of several million civilian entities, albeit of limited-functionality. They do not require either sophisticated behaviors or operator control. These are the so-called "clutter" entities that provide civilian entities, such as pedestrians, cars, motorbikes, *etc.* in the simulated environment.

.

2 DESIGNING FOR SCALABILITY

As previously mentioned, the JSAF/RTI-s application suite currently scales to over 300 federates and over a million entities (including simple clutter). However, current routing topologies limit the scalability of the overall system. In order for an interest-based communication infrastructure to scale, three conditions must hold over an arbitrary interval of simulation time:

- A client must generate a bounded number of messages
- A client must receive a bounded number of messages.
- Given the previous two points, the communication through any given router must also be bounded.

However, any given router must also be bounded, *i.e.* all-to-all communications are not conducive to scalability. An interest management system and careful federate design achieve bounded client communication. Bounded router communication is a function of network design and can be achieved using a mesh topology.

2.1 Interest Management

The aggregate amount of data produced by the JUO federation is greater than any one federate is capable of processing. An interest management system is used to limit the amount of data a federate must process (Rak, 1997). The federate declares which information it is interested in ("e.g., red force tanks in position cell X") and the RTI is responsible for ensuring only this subscribed information is received by the federate.

When used in a multicast environment, RTI-s utilizes the concept of multicast channels for filtering, with interest states having associated channels. The receiver filters the message at the kernel level, so the application never sees messages for interest states in which it is not interested. Due to the limited number of available multicast channels, the number of interest states is limited (increasing the amount of traffic associated with each interest state).

The ISI/Caltech team developed a more effective design. When running in point-to-point mode, interest management is "send-side squelched." Software routers maintain interest state vectors for each connection and only send messages to clients that have expressed interest in a message type. Because interest states are not tied to hardware and operating system limitations, the number of available interest states is comparatively unlimited. This is an enormous improvement over multicast IP. It was also one of the innovations of SF Express.

2.1.1 Routing Scalability

The scalability of the basic Mesh Router network is easily argued as follows. It is first necessary to assume that the underlying simulation problem itself has a scalable solution. This means a bounded message rate on the Primary $\Rightarrow$ PopUp and PullDown $\Rightarrow$Primary links within a basic triad, and bounded Up $\Rightarrow$ Down message rates within the interconnection links of the full network. The impediments to complete scalability of the mesh architecture have to do with interest declarations among the upper router layers. Each PullDown must announce its interest to every PopUp.

In principle, these interest broadcasts could be made scalable through an additional network of communication nodes. In practice, however, these interest updates were not frequent enough to cause any difficulties in SF Express or the JFCOM simulations with as many as thirty triads in the full mesh. An experiment with a similar mesh router setup using the current infrastructure shows similar results.

2.1.2 Routing Flexibility

The scalability issues with the tree router topology of RTI-s have been discussed previously. Tree topologies also map poorly onto physical wide-area networks. Figure 2, above, shows the route taken for any message crossing multiple sites in the JUO exercise. The path taken for a message to go from Maui to San Diego is sub-optimal: the data must first travel to Norfolk, then back to the west coast. This extra transmission time increases the latency of the system, which lowers overall performance. Since wide-area links often have less bandwidth available than local area networks, such routing also places a burden on the Virginia network infrastructure, which must have bandwidth available for both the incoming and outgoing message in the authors' Maui to San Diego example.

The mesh routing infrastructure provides a better utilization of physical networks by sending directly from one source to destination router. The network infrastructure is free to route messages in the most efficient way available. Figure 5 shows one possible routing topology for the JUO exercises, using mesh routers to minimize the distance messages must travel.

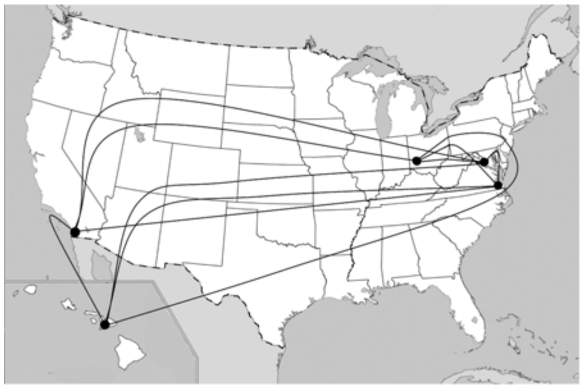


Figure 5: Advanced Routing Topology for JUO Exercises

In an ideal world, the entire federation would use one fully connected mesh for message routing. The actual routing of messages would be left to the physical network infrastructure. This Internet technology has over 30 years experience in optimizing data. However, such a configuration is often not feasible due to performance or protocol availability. Local area communication is usually over TCP, pushing error detection from RTI-s to the network stack. Over wide area networks, however, TCP suffers bandwidth degradation proportional to latency, so UDP is used for these connections. Some SPPs provide neither TCP nor UDP on computer nodes, instead providing MPI over a high-speed network, or provide public access only on a small subset of the machine.

The mesh router provides the ability to design a flexible network topology that meets the constraints of the network infrastructure while providing the ability to design a scalable system. The mesh router's topology is constructed by combining two building blocks: a tree (Figure 6, left) and a fully connected mesh (Figure 6, right). The two building blocks can be combined to form meshes of meshes, trees of meshes, meshes of trees, etc. The process can be repeated as often as required to build a suitable topology, but the mesh is scalable.

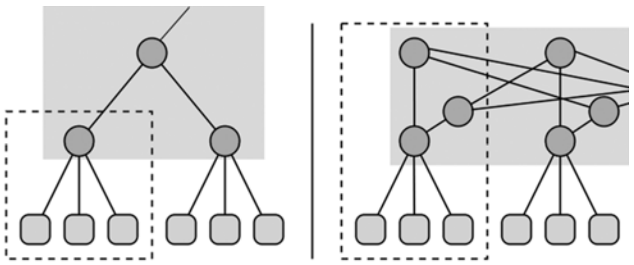


Figure 6: The Basic Building Blocks for a Mesh Router Topology: Tree (left) and Mesh (right)

2.2 Mesh Router Architecture

The mesh routers developed for RTI-s adopted many of the design decisions made in the SF Express project. The router triad concept is perhaps the most obvious of the design decisions from SF Express, providing an elegant method of avoiding "message looping" in the mesh, while allowing an arbitrary number of routing decisions to be made when transferring messages. However, significant design changes have produced a radically more advanced and flexible infrastructure.

2.2.1 Comparing Tree and Mesh Routers

In an analog of the decision to use HPC rather than just staying with workstation technology, one can easily decide that the straightforward Tree Router design is initially adequate, but eventually, one is confronted with the need to optimize. When the users demand new capabilities, more entities, or behavioral veracities, the potential benefits of the Mesh Router are not only desirable, they quickly become necessary.

2.2.2 Flow Control

A tight flow control with Request to Send / Clear to Send (RTS/CTS) behavior was used in the SF Express design.

SF Express used the mesh routers within SPPs, where latencies were extremely low and available bandwidth greatly exceeded expected message transfer rates. The overhead of sending the RTS and CTS messages would not negatively impact the performance or scalability of the system. The communication medium of choice (MPI) requires pre-posted receive buffers of a known size, requiring a RTS/CTS protocol for sending large messages. However, recent trends have shown CPU power improvements far outpacing network latency and bandwidth improvements. On modern networks, a RTS/CTS protocol poses a significant performance burden. Therefore, the Mesh Router architecture has an eager send protocol with messages dropped by priority when queues overflow.

2.2.3 Application-Independent "Message" and "Interest" Objects

The Mesh Router software is object-oriented (C++), with a limited number of standard interfaces to "user message" and "interest" base classes. For present purposes, the implications of this factorization are:

- The Mesh Router system is designed to be compatible with ongoing changes and evolution within the RTI-s system, requiring little more than "recompile and re-link".
- The Mesh Router system can support applications other than SAF/RTI, given appropriate different instances of the message and interest objects.

2.2.4 Factorized Communications Primitives

The Mesh Router object design relies on a very careful isolation/factorization of the underlying message exchange protocol from the rest of the JSAF software. The essential object design is conceptually indicated in Figure 7 and has three layers:

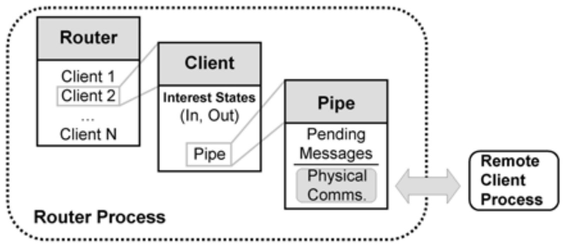


Figure 7: Schematic Design of the Mesh Router Application

2.3 Performance Testing Router Architectures

In a continuing effort to further quantify the relative utility of advanced router designs, new performance runs were accomplished in the first quarter of 2005. ASC-MSRC's Linux Cluster, Glenn, is basically the same configuration as Koa at MHPCC. In the case of the cluster at Wright-Paterson, there are 60GB hard disks on each local node, a configuration subsequently installed on MHPCC's Koa.

The implemented simulation utilized for the Glenn performance studies again used timed message exchanges between pairs of simple federates. One processor within each pair initiates a sequence of fifty fixed-size message exchanges with its partner, adding the times for each "there and back" message exchange. The process is repeated for a number of different message sizes. The primary output for each master-slave pair is simply a list of average exchange times versus message size.

A baseline set of measurements were taken using the Tree Router architecture. Runs were conducted at differing message sizes and communications were monitored between nodes that varied the number of Tree Routers through which they had to pass (hops) from one node to the other. Similar runs were conducted for the Mesh Routers, using the same terminology to describe the relative separation of the nodes on the mesh, even thought the actual hops required became a misnomer, as the Mesh Routers always pass through the same number of routers (Gottschalk, 2005)

Figure 8 presents the finding that MeshRouter 2-Hop and 3-Hop message delivery times are typically 2-4 times faster than the tree router in the 1Kbyte-10Kbyte message range, typical in JSAF. The bi-modal timing distribution indicates significant contention, as the individual messages are all pushed through a limited number of high-level routers. (Put differently, 20% of the communications pairs are left waiting while the first arrivals get out of the way). MeshRouter performance measures are remarkably insensitive to the Master/Slave separations. Note that with 0 hops, there is no significant difference between the architectures. As the size of the simulation grows, either due to span or complexity of the scenario, the advantage of the mesh routers becomes more evident. These differences persist up beyond 10,000 Byte messages.

TreeRouter performance degrades substantially as the underlying physical message path increases, the MeshRouter scales with point-to-point performance that is largely insensitive to the overall problem size. TreeRouter performance degrades substantially as the underlying physical message path increases, the MeshRouter scales with point-to-point performance that is largely insensitive to the overall problem size.

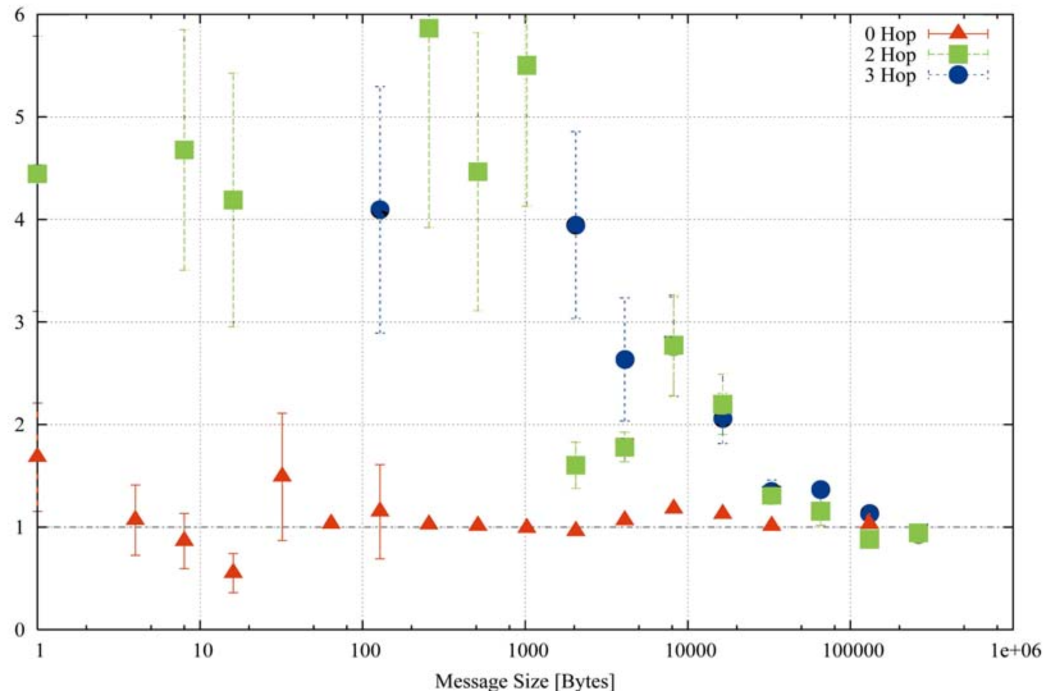


Figure 8: Relative Enhanced Throughput for MeshRouters versus TreeRouters

3 DATA COLLECTION AND MANAGEMENT

3.1 Baseline Plan for Processing Collected Data

The simulation would be of little and transitory use if the data collected could not be probed and studied. The initial design created by the JFCOM team used a centralized logging approach. In this design, a single logging process subscribes, through RTI interest management, to all messages published by the simulators. This all-to-one communication scheme performed reasonably on local area networks with a small number of simulators. But this approach does not scale to hundreds of simulators running on multiple clusters. In such computing environments, the centralized logger becomes the bottleneck. The physical network and routers cannot keep pace with the volume of messages. They are forced to drop simulation messages. Even if the network was able to deliver the messages to the centralized logger, the logger will not be able to store the messages and to process them for analysis in a timely manner.

The initial implementation of the design was limited by the use of the Access2000® database, which had an internal limit of 2GB. This forced the collected data to be stored in separate data segments. After a simulation run was completed, the collected database segments were processed into the single MySQL database in the following sequence: First, a database schema was applied to the MySQL database that would represent the “rollup” of all of the data collected. Second, the Access2000 database segments were put into sequential order by their unique file names and internal file creation timestamps. Third, each segment was filtered to build a list of relevant tables from which data is to be extracted. Fourth, the first Access2000 database was inserted directly into the MySQL database. Fifth, subsequent Access2000 databases were processed to ignore overlapping data by examining the difference in timestamps between where the previous segment ended and the next segment began. Also, internal record timestamps were adjusted with an offset so that individual record timestamps represented time since beginning of simulation run and not just for the individual database segment. Finally, summarization information was extracted and stored into new tables to facilitate speed in reviewing common information. Those reports that were processing-intensive will be generated and saved for post-event review.

3.2 Post-Event Data Processing

Once the simulation data had been processed and inserted into the MySQL database, the MOP/MOE (Measure of Performance/Measure of Effectiveness) tools were applied to the completed database to provide predefined statistics for the event period. In conjunction with these predefined reports, additional reports and queries can be rapidly created based on additional feedback and desires of the analyst. A sample of predefined reports available for the end user included: Killer/Victim scoreboard, Entity Lifecycle and Lifecycle Details, and "String" analysis charts. Other MOP/MOE components were developed and included with the toolkit based on input from the data collection and analysis plan designed by the joint experimentation users and analysts.

3.3 Implementation of Enhancements

The ISI/IDA data team made changes to the baseline plan for data collection and analysis as new challenges arose along the way. The key challenge is to provide the capability to log all simulation messages without adversely affecting and interfering with the performance of the JSAF simulation. The approach taken is to implement a distributed logger that minimizes network communication overhead by logging the data at the data generation source, and by selectively propagating that data, based on need.

Instead of one centralized logger, a local logger was created for each simulator. Conceptually, this local logger intercepts messages emitted by the local simulator to the RTI communication layer. The intercepted messages are stored locally on the compute node. No network logger traffic is generated during the logging process. As long as each local logger can keep up with the local simulator, this approach provides seemingly perfect scalability with respect to the number of simulators. As more simulators are added more local loggers are added. There seem to be no centralized bottlenecks in this approach.

3.3.1 Interceptor/Logger

The Interceptor/Logger application is a process that resides on individual simulation nodes within the federation. The determining factor on where to utilize the mechanism is determined by which federates are publishing information needed for data collection. The interceptor/logger, utilizing functionality in the RTI Application Programming Interface, inserts "hooks" into the published data streams by the RTI and then splits off two child processes, one process that writes and compresses the intercepted data into binary "log" files. A second process is one that decodes the data stream and inserts the decoded data into an embedded database application, initially SQLite, now MySQL. A separate daemon process called "sqlited" handles incoming socket-based connection attempts to query information that has been stored in the local database. Figure 9 is a diagram of the process.

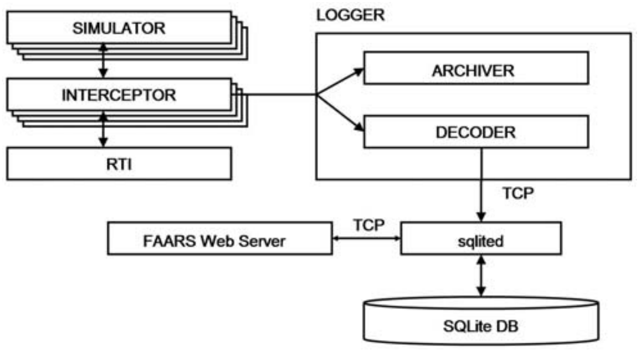


Figure 9: Interceptor/Logger Process

Because of the methodology of running interceptor/loggers on each simulator with data of interest, a separate mechanism was needed to retrieve information stored at each simulator location. A separate application process called "Aggregator" was developed that would handle the intercommunication between simulators logging data. The Aggregator is configured in a tree-like fashion, with a "Root Aggregator" at the head of the tree and "Child Aggregators" in branches from the root. The various branches reach out to the individual leaf instances of "sqlited" on each simulator. The interface to the Root Aggregator takes a Structured Query Language-formatted query and passes the query on to each of the branch Child Aggregators until the query finally reaches the individual instances of "sqlited". As each instance of "sqlited" responds with the requested data for the query, the Child Aggregators assemble the returned information in order of response and forward the data on to the Root Aggregator, which then assembles the complete, returned information and forwards it on to the original requestor. The Aggregator model works on Transmission Control Protocol (TCP) socket-based connections between the Root Aggregator and subsequent Children Aggregators.

3.3.2 Near Real Time Retrieval Of Data

With the utilization of the ISI interceptor/logger, the possibility of retrieving simulation information in a "near real time" manner became a reality. Typically, data collection efforts have had to wait until hours after collected logger files have been processed before any specific event information could be derived. This is a vast improvement in functionality and provides a wide range of uses that are still being realized as we move forward in the software development effort.

The Near Real Time data retrieval effort is based around the ability to query the ISI interceptor/logger application, retrieve the logged information from each node and store the retrieved information into a local Relational Database Management System (RDBMS). The retrieved information is then used by the FAARS Near Real Time web server interface to allow users of the system to view various reports, charts and graphs based on the available information.

The process of retrieving intercepted information from each of the active ISI interceptor/loggers is handled by a series of BASH shell scripts on the FAARS web server. Each BASH shell script is targeted towards retrieving specific information, such as entity object states, and is used to process the retrieved information into the local RDBMS (aka cache). The data retrieval process is based on three steps. The first step is to send the request for information to the Root Aggregator. The methodology used by the retrieval process is based on making a TCP socket-based connection to the Root Aggregator and sending an SQL-formatted query. The second step is to wait for a response and process/validate the retrieved data and write this data to a temporary file. The response from the Root Aggregator is a stream of plain ASCII text, which is tab-delimited for fields and is carriage return delimited for individual records. This information is then written to a temporary file in “tab-delimited/carriage return delimited” format. The third and final step is to load the temporary file's data into the local cache.

The FAARS web server RDBMS cache now uses MySQL (v4.1.1.) as the database engine. The database schema for the cache is based primarily on the schema used by the ISI interceptor/logger. This helps in facilitating compatibility with the information that is being utilized in near real time and data being reviewed post event. The main difference between near real time and post event processing is the different indexing schemas utilized on the local cache. The indices applied to the local cache database have been specifically tuned to support the types of queries that the FAARS web server uses for data displays.

Because of the nature of distributed logging of the data that is currently implemented, it is now necessary to develop means to:

- retrieve all of the saved binary data logs on each simulator, with the ISI interceptor/logger
- prepare and decode the binary data files
- insert the decoded data into a consolidated database (the complete accumulation of data for a that event.)

A process called "Data Staging" has been developed that accomplishes these tasks in an organized, efficient manner, making the best usage of available bandwidth, processing cycles and disk space. The Data Staging process begins with retrieving the binary log files at the end of each day's simulation run from each simulator logging data. The data is moved and stored on the local storage point in a hierarchical format based on the event name, day of the event and the simulator where the log file was retrieved. Once the data has been moved, Perl-based scripts are run against the individual binary log file to decode and format the binary data into plain-text, comma-separated value (CSV) flat files. The translation of the data and the creation of the storage database schema are based on utilizing definitions found in the Federation Object Model (FOM) and Federation Execution Document (FED) for the federation in use. Each CSV-formatted file represents a section of data to be inserted into the consolidated database for the event. A final Perl-based script takes the CSV-format files and inserts the decoded data into the appropriate table within the consolidated database.

4 ACCOMPLISHMENTS AND FUTURE DIRECTIONS

4.1 Accomplishments

In December of 2002, the JESPP team ran a successful prototype event using a partition of the USC Linux cluster, consisting of some 240 servers, with 2 GHz Xeons, 1 GByte of RAM and both GigE and Myrinet mesh communications. The scientists at ISI in California and the operators at JFCOM in Virginia jointly shared control. More than 1,000,000 civilian entities were successfully simulated. They showed appropriate behavior and were stable, even when scanned by the SLAMEM program, emulating two GlobalHawk platforms. To ensure usability and operational validity, about 1,100 warfighting entities were also simulated and controlled in a manner consistent with normal J9 experimentation. Stability and appropriate response to control commands were evident throughout. Several runs were conducted over the course of a week and performance was characterized.

Following the December event, it was decided to show the utility of the DoD’s SPP assets by using two Linux clusters, at two High Performance Computing Modernization Program sites, MHPCC and ASC-MSRC. The HPCMP then established a new Distributed Center (DC) to provide computing assets for this work. The senior staffers of the HPCMP were most helpful in discussions relating to the appropriate hardware for the task at hand. As an illuminating example, they raised the issue as to the potential benefits of installing 64 bit AMD or Intel processors, but were very sensitive to the ultimate customer’s desire that the leap to high-end computing and parallel processing on Linux clusters remain as closely compatible with the existing Linux workstation configurations. The system was installed in the spring of 2004 and became operational within weeks.

Issues including the location of the Linux cluster, government security, networking bandwidth and latency, and cluster capabilities were surfaced and resolved. The desire for both redundancy and the desirability of further proving distributed high performance computing militated in favor of splitting the 256 nodes (two processors, 4GB RAM per node) into two machines, one at the MHPCC in Hawai'i and one at the ASC MSRC in Ohio. Both of these goals have been met and the original analysis has been proven prudent many times, when one site was able to carry the entire load while the other was faced with some type of outage, administrative encumbrance (*e.g.* Center power system maintenance) or network isolation.

Networking turned out to be a small factor and evidenced itself in unexpected ways. The underlying simulation program, one of the SAF family, tolerates latencies quite well. Even latencies up to 500 milliseconds (0.5 seconds) are tolerable. However, the speed-of-light-latency alone from Virginia to Maui is on the order of 100 milliseconds, and that was disorienting for the operators, not in the performance of the simulation, but in graphical user interface (GUI) issues like the delay in a drawn circle responding to mouse/cursor movements.

4.2 Open Issues for Future Work

There is much to be done in terms of instrumenting and analyzing the existing system, contrasting performance with that from communications options within the current RTI-s baseline. The more interesting studies here will involve comparisons of new qualitative features of the underlying simulations. An example is the difference between “reduced capability” and “self-aware” clutter (*i.e.*, do clutter objects interact). Many of the more interesting near-term development paths can be characterized in terms of “special purpose gateways”.

5 CONCLUSION

While the techniques proposed in this paper may not be a universal panacea, the authors maintain those techniques will increase the opportunity for the defense analysts to discover new and dangerous threats and work out the best way to defend against the destruction from those threats.

The mesh router infrastructure presents a scalable routing infrastructure for both local and wide area communication. They are capable of being organized into a number of topologies and should be easily extensible. For wide area networks, the flexible routing topologies allow communication over all available network links, without the hub and spoke problem of the tree routers. Within a local area network, the mesh routers provide a scalable communication architecture capable of supporting hundreds of federates.

This was not merely a translation of existing communications procedures. It was the first of a number of steps to achieve the qualitatively new capabilities that follow from the scalable communications of the basic architecture. It will also provide capabilities of the “intelligent gateways” for WANs, supportable within this architecture.

ACKNOWLEDGMENTS

The authors would like to thank the Joint Experimentation Group at JFCOM and they acknowledge the support and encouragement of Major General Woods, Anthony Cerri, and Jim Blank. The authors especially would like to acknowledge the direction, support and counsel of Rae Dehncke, program manger and guiding light for this project. They would like to recognize the excellent technical support of the ISI’s Computational Sciences Division, especially John Tran and the guidance and assistance from ISI’s Scalable Systems Division and the JSAF team: Dr. Andy Ceranowicz, Mark Torpey, Bill Hellfinstine and many others. This material is based on research sponsored by the Air Force Research Laboratory under agreement numbers F30602-02-C-0213 and FA8750-05-2-0204. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes, notwithstanding any copyright notation thereon.

REFERENCES

Barrnett, T.P.M. 2004. *The Pentagon’s New Map: War and Peace in the Twenty-First Centur*y, New York, New York, Putnam Company.

Ben-Ari, E. 1998. Mastering soldiers: conflict, emotions and the enemy in an Israeli military unit, *New Directions in Anthropology*, V. 10. Oxford: Berghahn Books.

Brunett, S., D. M. Davis, T. D. Gottschalk, & P. Messina. 1998. Implementing distributed synthetic forces Simulations in metacomputing environments, *Seventh Heterogeneous Computing Workshop*, Orlando, Florida.

Brunett, S., and T. D. Gottschalk, 1997, Scalable ModSAF simulations with more than 50,000 vehicles using multiple scalable parallel processors, Technical Report CACR-156, 1997, Caltech, presented at *Spring Simulation Interoperability Workshop*, Orlando, Florida.

Calder, R. B., J. E. Smith, A. J. Courtemanche, J. M. F. Mar, and A. Z. Ceranowicz. 1993. ModSAF behavior simulation and control. In *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation. Orlando, Florida: Institute for Simulation and Training*, Univ. of Central Florida.

Cebrowski, A.K., and J.J. Garstka, 1998. Network centric warfare: its origin and future, *Naval Institute Proceedings*, 124/1, 28-35, Annapolis, Maryland.

Ceranowicz, A. Z., M. Torpey, W. Hellfinstine, J. Evans, and J. Hines. 2002. Reflections on building the joint experimental federation, *Proceedings of the 2002 I/ITSEC Conference*, Orlando, Florida.

Ceranowicz, A. and M. Torpey. 2004. Modeling human behaviors in an urban setting, *Proceedings of the 2004 I/ITSEC Conference*, Orlando, Florida.

Dahmann, J., J. Olszewski, R. Briggs, and R. Weatherly. 1997. High Level Architecture HLA performance framework, *Fall 1997 Simulation Interoperability Workshop*, Orlando, FL, 1997.

Davis, D. M., G. D. Baer, and T.D. Gottschalk. 2004. 21st Century simulation: exploiting high performance parallel computing and advanced data analysis, *Proceedings of the 2004 I/ITSEC Conference*, Orlando, Florida

Defense Modeling and Simulation Office. 1998. *High Level Architecture Interface Specification*, v1.3, 1998.

Foster, I. and C. Kesselman 1997. Globus: a metacomputing infra-structure toolkit, *Intl Journal Supercomputer Applications*, 112: 115 –128

Fujimoto, R. and P. Hoare. 1998. HLA RTI performance in high speed LAN environments, in the *Fall Simulation Interoperability Worksho*p, Orlando, Florida.

Garlan, D. and M. Shaw. 1993, An *Introduction to Software Architecture: Advances in Software Engineering and Knowledge Engineering*, volume I. New York, New York, World Scientific Publishing.

Gottschalk, T. D, P. Amburn, and D. M. Davis. (Forthcoming, 2005), "Advanced message routing for scalable distributed simulations," *The Journal of Defense Modeling and Simulation*, San Diego, California, Publication pending

Hill, R. W., J. Gratch, and P.S. Rosenbloom. 2000. Flexible group behavior; virtual commanders for synthetic battlespaces. *Proceedings of the Fourth International Conference on Autonomous Agents*, Barcelona, Spain.

Kaufman, W. and L. Smarr. 1993. *Supercomputing and the Transformation of Science*, New York, Scientific American Library

Keahey, K. and D. Gannon. 1997. PARDIS: A parallel approach to CORBA, *Proceedings. The Sixth IEEE International Symposium on High Performance Distributed Computing*, pp: 31-39, Portland, Oregon

Lucas, R. F. & Davis, D. M. 2003,. Joint Experimentation on Scalable Parallel Processors. *In Interservice/Industry Training, Simulation, and Education Conference*, Orlando, Florida.

Messina, P. C., S. Brunett, D. M. Davis, and T.D. Gottschalk. 1997. Distributed interactive simulation for synthetic forces, In J. Antonio, Chair, *Mapping and* Scheduling Systems, *International Parallel Processing Symposium*, Geneva, Switzerland.

MPI Forum 1993. MPI: A message passing interface. In *Proceedings of 1993 Supercomputing Conference*, Portland, Washington.

Rak, S., M. Salisbury, and R. MacDonald. 1997. HLA/RTI data distribution management in the Synthetic Theater of War, *Proceedings of the Fall 1997 DIS Workshop on Simulation Standards*, Orlando, Florida

Sanne, J. 1999. *Creating Safety in Air Traffic Control*. Unpublished doctoral dissertation, Institute of Tema Research, Linköping University, S-581 83 Linköping, Sweden.

van Lent, M., and K. Laird, 1998. Learning by observation in a complex domain. *Proceedings of the Knowledge Acquisition Workshop*, Banff, Canada.

AUTHOR BIOGRAPHIES

GENE WAGENBRETH is a Systems Analyst for Parallel Processing at the Information Sciences Institute at the University of Southern California, doing research in the Computational Sciences Division. He specializes in tools for distributed and shared memory parallelization of Fortran programs and has been active in benchmarking, optimization and porting of software for private industry and government labs. <genew@isi.edu>

KE-THIA YAO is a research scientist in the Distributed Scalable Systems Division of the University of Southern California Information Sciences Institute. Currently, he is working on the JESPP project, which has the goal of supporting very large-scale distributed military simulation involving millions of entities. Within the JESPP project he is developing a suite of monitoring/logging/analysis tools to help users better understand the properties of large-scale simulations. <kyao@isi.edu>

DAN M. DAVIS is the Director, JESPP Project, Information Sciences Institute (ISI), University of Southern California, and has been active for more than a decade in large-scale distributed simulations for the DoD. While he was the Assistant Director of the Center for Advanced Computing Research at the Caltech, he managed Synthetic Forces Express, a multi-year simulation project and pursued other simulation interests. <ddavis@isi.edu>

ROBERT F. LUCAS is the Director of the Computational Sciences Division of ISI at the University of Southern California. There he manages research in computer architecture, VLSI, compilers and other software tools. He has been the principal investigator on the JESPP project since its inception in 2002. <rflucas@isi.edu>

THOMAS D. GOTTSCHALK is a Lecturer in Physics at the California Institute of Technology and a Member of the Professional Staff, Center for Advanced Computing Research there at Caltech. His research has been in the field of the use of parallel computers to simulate various physical phenomena. <tdg@cacr.caltech.edu>

Appendix E

Developing Situation Awareness Metrics in a Synthetic Battlespace Environment

Jacqueline M. Curiel, Michael D. Anhalt
Alion Science and Technology
Fairfax, Virginia
{jcuriel,manhalt}@alionscience.com

John J. Tran, Ke-Thia Yao
Information Sciences Institute/USC
Marina del Rey, California
{jtran,kyao}@ISI.EDU

ABSTRACT

The Joint Forces Command (JFCOM) conducts Joint Urban Operation (JUO) exercises in synthetic battlespace using human-directed computer simulation tools such as Joint Semi-Automated Forces (JSAF) to support ongoing joint war-fighting efforts. A component of these experiments is that of human-in-the-loop (HITL) interactions where human players impact the outcome of the exercise. This is in contrast to Monte Carlo constructive experiments that only involve computer behavior. The need to objectively measure the effectiveness of human players and their interaction with the simulation environment requires quantitative metrics to supplement more qualitative observer-based judgments. Situation awareness (SA), a cognitive behavior captured in HITL experiments, involves the perception and comprehension of forces and events in a situation, and a prediction of their future status, Endsley (1995). Objectively measuring SA is drawing intense interest because this knowledge is crucial to successful decision-making processes (C2).

Building upon work presented at I/ITSEC 2004 (An Interdisciplinary Approach to the Study of Battlefield Simulation Systems, paper 1886), we adopt a cognitive-computational approach for measuring SA based on Situation Model theory. Situation models are complex mental representation of events. As events unfold, these mental representations must be updated to maintain an accurate representation. Prior research has demonstrated that situation models are updated along a number of dimensions. These dimensions reflect information about entities, space and time coordinates, participants' goals, and the causal relationships of events. We utilize the information encapsulated in SA objects (SAOs), recorded during the JUO exercises, to develop a tool that automatically monitors players' SA and evaluate the importance of these dimensions on situation awareness over the time course of the experiment and on the three levels of SA. Our findings have practical implications for subsequent training, product development, and extend the knowledge base of cognitive behavior.
.

ABOUT THE AUTHORS

Jacqueline M. Curiel is a research psychologist at Alion Science and Technology. She is also a co-founder of Behavioral Cognition and is a consultant to IdeaDaVinci, a technology incubator. Her prior academic experience includes teaching and research positions at the University of Texas at San Antonio and the University of Notre Dame, where she did her graduate work. Her published work has primarily included work on mental maps and situation models, the focus of her Master's thesis and doctoral dissertation.

John J. Tran is a researcher at the Information Sciences Institute, University of Southern California. He received both his BS and MS Degrees in Computer Science and Engineering from the University of Notre Dame, where he focused on Object-oriented software engineering, large-scale software system design and implementation, and high performance parallel and scientific computing. He has worked at the Stanford Linear Accelerator Center, Safetopia, and Intel. His current research centers on Linux cluster engineering, effective control of parallel programs, and communications fabrics for large-scale computation. Capt Tran is also a member of the 129th Rescue Wing at Moffett FAF, California.

Michael D. Anhalt is retired Navy Surface Line Commander with over 23 years of operational experience, including specialties in Amphibious Warfare, Surface, Undersea, and Strike Warfare, and tactical training. Twelve years experience in planning and directing system-engineering efforts related to modeling & simulation and their integration with military command and control (C2) systems. Provides on-site technical support in planning for and conducting warfighting exercises and experiments, prototype development, and demonstration of advanced technologies for next generation C2 Systems and Command Centers. He holds a Master of Science degree in Educational Technology.

Ke-Thia Yao is a research scientist in the Distributed Scalable Systems Division of the University of Southern California Information Sciences Institute. Currently, he is working on the JESPP project, which has the goal of supporting very large-scale distributed military simulation involving millions of entities. Within the JESPP project he is developing a suite of monitoring/logging/analysis tools to help users better understand the computational and behavioral properties of large-scale simulations. He received his B.S. degree in EECS from UC Berkeley, and his M.S. and Ph.D. degrees in Computer Science from Rutgers University.

INTRODUCTION

Problem Description

The “one-the-fly” nature of large-scale human-in-the-loop (HITL) experiments, such as those supported by the Joint Semi Automated Forces (JSAF) simulation federation, mirrors that of actual warfare. The scenarios played out in these types of experiments reflect the continuous interaction among forces (i.e., friendly, hostile, and neutral) over the time course of the experiment so that the situation is dynamic, unfolding over time. These aspects of HITL experiments constrain both the players’ capabilities of maintaining accurate Situation Awareness (SA) and the evaluators’ attempts to assess players’ SA in an effective and timely manner.

The problems associated with assessing SA indicate an interest in further understanding the processes involved in situation awareness during these types of experiments and the continued development of performance metrics. Currently, in HITL experiments, players use sensors to detect the presence of entities and their location, which is necessary for situation awareness but not complete. Additionally, SA depends on identifying the proper context of the experiment. This paper presents our current efforts to develop SA metrics.

Motivation

The motivation for this paper is twofold: 1. previous HITL experiments have yielded a wealth of information that is readily available and, for our purposes, provide a useful base to develop our metrics and 2. current methods of evaluating SA in these types of experiments include observations of players during the exercise and players’ reports afterwards. Both measures tend to be subjective, making it more difficult to identify and break down different aspects of situation awareness. We believe that incorporating what we know about situation awareness and situation models with the existing data will help us develop metrics that will help us better understand players’ situation awareness.

SITUATION AWARENESS AND SITUATION MODELS

The numerous uses of situation(al) awareness underscore its popularity in research applications. Situation awareness includes an awareness of friendly and enemy troop positions at a specific point in time (Pew & Mavor, Eds. 1998). Another more specific view of SA, Endsley’s (1998) three level approach has enjoyed widespread acceptance and has been used in numerous research endeavors to investigate SA. Of interest here is its use in evaluating player performance.

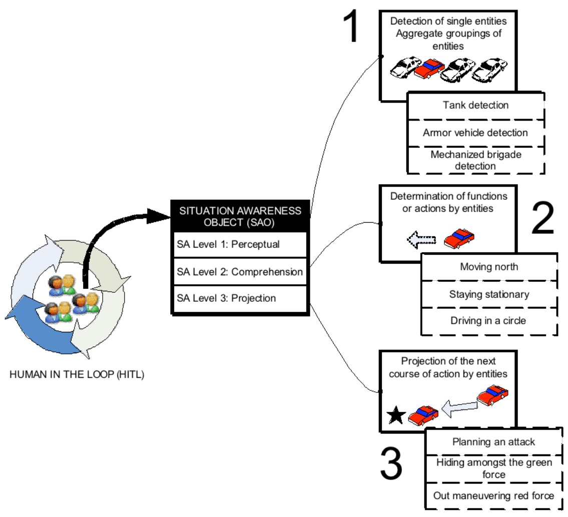

HUMAN IN THE LOOP (HITL)
SITUATION AWARENESS OBJECT (SAO)
SA Level 1: Perceptual
SA Level 2: Comprehension
SA Level 3: Projection
1
Detection of single entities Aggregate groupings of entities
Tank detection
Armor vehicle detection
Mechanized brigade detection
2
Determination of functions or actions by entities
Moving north
Staying stationary
Driving in a circle
3
Projection of the next course of action by entities
Planning an attack
Hiding amongst the green force
Out maneuvering red force

Figure 1. Endsley’s SA model specific to synthetic battlespace environment

According to Endsley, SA can be described as three interdependent levels corresponding to: (1) Perceptual SA, (2) Comprehension SA, and (3) Projection SA (Figure 1). The perceptual level involves the detection, recognition, and identification of elements that define a specific situation. Perceptual SA relies on available sensory information, (e.g., from sensors in the case of a player in a HITL experiment) and the player’s prior knowledge (e.g., object patterns/schemas activated in memory) to identify individual situation elements and object groups, based on their characteristics.

Comprehension SA reflects an understanding of the current state of affairs and involves making inferences about activities in the current situation. As such, the comprehension level maps the products of perception to object functions. Finally, projection SA consists of interpretations about the trajectory of the situation based on the products of Comprehension SA and prior knowledge. These interpretations include identifying the range of possible trajectories or courses of action along with determining the likelihood of occurrence of each.

At all SA levels is affected by uncertainty due to a number of factors, such as limitations of sensors, and limitations in player's prior knowledge, and the goals of the enemy. Figure 2 shows questions that are relevant to all three SA levels.

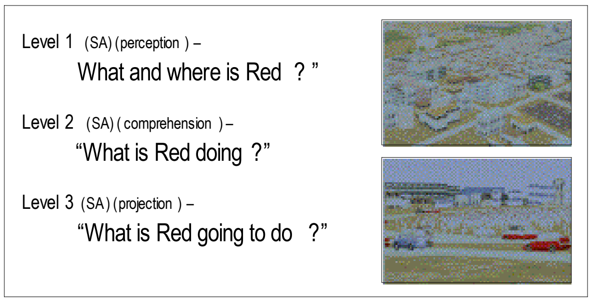

Level 1 (SA) (perception) –
What and where is Red ? "
Level 2 (SA) (comprehension) –
"What is Red doing ?"
Level 3 (SA) (projection) –
"What is Red going to do ?"

Figure 2. Desired SA Level Metrics in JUO

In our 2004 I/ITSEC paper [1866, Tran, Curiel & Yao], we proposed that the findings of reading comprehension experiments used to study situation models could guide the evaluation of situation awareness in JSAF HITL experiments. Situation Models, mental representations of a situation, are analogous to the mental products of Comprehension SA. Likewise, these representations also depend on the products of lower levels of processing (e.g., textbase and propositional representations in the case of reading comprehension) as well as prior knowledge (e.g., situation schemas).

Zwaan and Radvansky's Event-Indexing model, have focused on providing empirical support for the idea that situation models are multi-dimensional. Although it is unclear how many dimensions can be involved, influences of space, time, entity/protagonist, causality, and intentionality have been observed (e.g., Zwaan & Radvansky). The findings have been interpreted as indicating that readers construct situation models that are defined by these dimensions and updated when changes in the situation occur. Once the story has ended, readers have encoded a completed situation model that is analogous to the "global static summary," an analysis of the end result of the HITL experiment in which the effectiveness of the strategy, the goals of the mission, and the effectiveness of the information provided by the sensors are evaluated. This paper focuses on the relationship between situation awareness and situation model dimensions in HITL a experiment.

EXPERIMENTAL BACKGROUND

Our focus is on the first phase of the Joint Urban Operations (JUO) Urban Resolve experiment conducted by the USJFCOM J9 Directorate and Joint Advance Warfighting Program (JAWP) to guide the development of future sensor capabilities that help soldiers fight in complex urban environments (Ceranowicz & Torpey, 2004**.** Urban Resolve Phase 1 focused on evaluating the use of human and advance intelligence, surveillance, and reconnaissance technologies to gain situation awareness. Future phases will focus on evaluating the ability to precisely shape the urban battlespace using advanced concept of operations.

Urban Terrain JUO Urban Resolve uses detailed high-fidelity entity-based simulations of urban city areas to exercise proposed sensor capabilities. The Urban Resolve terrain database includes dense urban road networks with over 1.8 million buildings (Prager et al., 2004). Some of these buildings are based on actual real world building footprints, and some have interiors to model parking garages. The terrain features includes elements like parked cars, dumpsters, jersey barriers, individual trees, tree canopies and trashcans. The terrain landscape ranges from deep urban canyons with tall buildings to flat parking areas and open spaces.

This urban terrain is inhabited by approximately 100,000 clutter entities (Speicher & Wilbert, 2004, Williams & Tran 2003). These clutter entities can range from ground vehicles to pedestrians to air/sea vehicles. At the individual entity scale, the ground vehicles follow traffic rules and behave properly at road intersections. At the aggregate scale, the ground vehicles follow the normal flow a bustling city. Rush hours occur during the morning and late afternoon as entities go to and from work. During the lunch hour people go on errand runs, and during the evening people go to restaurants.

Red Force Hiding within this urban terrain is the Red Force (Haskell et al., 2004). The Red Force primarily consists of dismounted infantries, but they also include heavy crew-served weapons, "technicals" (vehicles armed with heavy weapons), light transportation trucks, short-range air defense forces and artillery support. The Red Force follows Techniques, Tactics and Procedures (TTPs). The Red Force tried to blend in the urban environment by pretending to be part of the civilian clutter population, moving about the city to set up fighting locations by fortifying builds and creating booby traps.

Blue Force The objective of the Blue Force is to gain situation awareness of the Red Force. For UR Phase I, the Blue Force is made up of only sensors. The sensors include unmanned aerial vehicles (UAV), low flying organic aerial vehicles (OAV), unattended ground sensors (UGS) and human intelligence. The job for the Blue Cell human players is to task these sensors. observe and track the Red Force. Each Blue Cell human player is given access to a JSAF graphical map display. The map contains the detailed urban terrain overlaid with the positions of the sensors and the Red Force tracks generated by the sensors. The sensors are not completely accurate. The tracks may misclassify the Red entities, and the perceived entity location/velocity may vary from the actual location/velocity.

Procedure

Data for our analyses was obtained from the Urban Resolve Phase 1 set of experiments, which explored new approaches to urban combat. The general procedure follows below.

Participants The Blue Team was comprised of nine active- and reserve-duty military personnel, along with retired military and other contractors. They were selected for the experiment based on previous intelligence experience, or their command and control background, as well as for their ability to adapt to and use new software applications.

Pre-Experimental Training The blue team was given several weeks of training to enable them to become more familiar with application operations, such as the JSAF simulation system and IWS (Information Work Station), to provide briefings about projected enemy capabilities and their likely courses of action, and to provide intelligence briefings to help the players understand their dynamic activities.

Method The Blue Team occupied a room with computers and projected displays. Their main objective was to use their futuristic 2018 sensors to gain situation awareness of the Red Force by controlling their sensor placement and moving them as necessary to follow or anticipate enemy movement. Each player operated a command and control suite, made up of the JSAF simulation system, with two monitors that displayed a map and allowed for simulation control. They also used a collaborative tool application named Information Work Station (IWS) for chat, email, document sharing and discussions. During the trials, players communicated using Situational Awareness Objects, which recorded players situation information about the enemy, shared map overlays, Voice over Internet Protocol (VoIP), NetTalk chat, and limited face-to-face communications.

The players were briefed prior to each trial regarding enemy capabilities, activities and their likely courses of action. They were told what their sensor limitations were, based on the trial conditions and briefed on any modifications to the JSAF software that might affect their play. The team was flexible in establishing each member's responsibilities and over time, the team decided to have a Commander, with a Sensor Manager and a Surveillance Manger working directly for him. Six Sensor operators worked directly for the Sensor Manager, making sensor asset requests to the Sensor Manager.

Experimental Trials Along with the baseline trial, there were six experimental trials as can be seen in table 1. The type, numbers and capability of the sensors were modified for each trial to determine the impact of the specific changes in the resultant SA. Each trial lasted four or five days, with game play lasting about 7 hours.

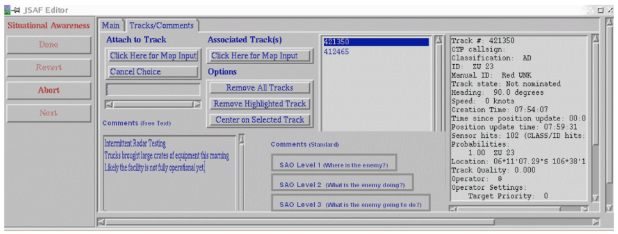
Table 1. Experimental Trials

Trial	Conditions	Duration (hours)
1	Base Case	48
2a	Inactive Red	24
2b	Poor Weather & Inactive Red	18
3a	Signature Reduction	24
3b	½ Inventory	24
4a	No Tags	24
4b	No Tags, ½ Inventory, Signature Reduction	24

For Trial 1, which served as the base case scenario, players had full use of all sensors and the enemy was on the move. For Trial 2a, the enemy moved less frequently and therefore had less exposure to the sensors. For Trial 2b, cloud cover obscured the high altitude sensors and so that there was less initial detection. For Trial 3a, the enemy was allowed to use camouflage. For Trial 3b, the number of sensors was reduced by half. For Trial 4a, futuristic radio frequency (RF) tagging of vehicles and humans was not used, and the enemy could not use camouflage. For Trial 4b, the sensor inventory was reduced by half, the enemy could use camouflage and players could not use RF tagging.

SAO Objects

Our data were obtained from Situation Awareness Objects (SAOs), a method of recording information about red force entities that has only been used this series of experiments. The SAO is a compact package of information that players create and place on a shared terrain map that contains their thoughts, assumptions, and their understanding regarding the enemy. The SAOs are created by selecting options from pull down menus tailored for the trial and modified as the players requested more options. The SAO includes an option to let the players include free-text. Figure 3 shows the SAO screens and sample comments.



Intermittent Radar Testing
Trucks brought large crates of equipment this morning
Likely the facility is not fully operational yet

```
Track #: 421350
CTP callsign:
Classification:  AD
ID:  ZU 23
Manual ID:  Red UNK
Track state: Not nominated
Heading:  90.0 degrees
Speed:  0 knots
Creation Time: 07:54:07
Time since position update: 00:0
Position update time: 07:59:31
Sensor hits: 102 (CLASS/ID hits:
Probabilities:
    1.00  ZU 23
Location: 06*11'07.29"S 106*38'1
Track Quality: 0.000
Operator:  0
Operator Settings:
    Target Priority:  0
```

Sample Blue standard comments (by SAO category as prescribed by the user)

Level 1	Level 2	Level 3
Appears to be a mortar site	Appears to be a group of insurgents placing a mortar pad	Expect a launch at sunset
Enemy Recon team on roof	Recon team is waiting until street clears	Recon Team will radio mortar team when all clear
Vehicles and Explosives in bldg	Explosives are being loaded into vehicles	Vehicles likely to move to FOB 3 during the night

Sample Red standard comments (by SAO category as prescribed by the user)

Level 1	Level 2	Level 3
Establishing a mortar site	Insurgents placing a mortar pad	Plan a launch at sunset
Positioning a Recon team on roof	Recon team is waiting until street clears	Recon Team will radio mortar team when all clear
Moving Vehicles and Explosives into bldg	Explosives are being loaded into vehicles	Vehicles will move to FOB 3 during the night

Figure 3. SAO input screen

SAOs allow players to quickly enter relevant SA data during the experiment and are shared among other players dynamically and instaneously amongst all the players. They support two complementary objectives: team collaboration and data collection for after action review and data analysis. SAO options are designed to be comprehensive, but not to have players decide the level of SA they refer.

The use of SAOs supplement existing techniques used to assess situation awareness and reduce the analyst's need to intrude on the player's activities in order to assess their performance.

Figure 4 shows that the SAOs are tailored to provide players with relevant real-time data to support their understanding and assessment of the player's SA. Further, SAOs can be used for more in-depth analysis after the experiment trial, allowing analysts to compare actual enemy activities with the SAOs. The SAO approach is successful because the players gain benefit from using SAOs allowing them to share information rapidly and SAOs provide a resource for the analysts to easily and rapidly assess player SA.

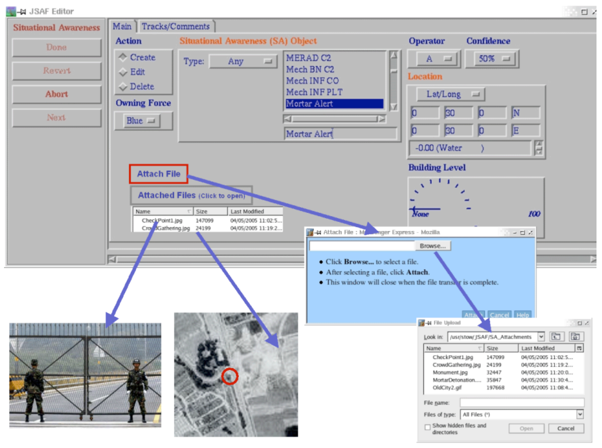


Figure 4. SAOs map to real life model

ANALYSIS

Figure 5 shows counts of the SAO comments for each trial. As can be seen, the Baseline Trial showed the most SAO comments, which is not surprising, given that the duration of that trial was at least twice as long as the other six trials. Of note is that in Trial 4b, which did not use the futuristic RF tags and had both ½ inventory and signature reduction, showed slightly more SAO comments than either the signature reduction trial (3a) or the ½ inventory trial (3b).

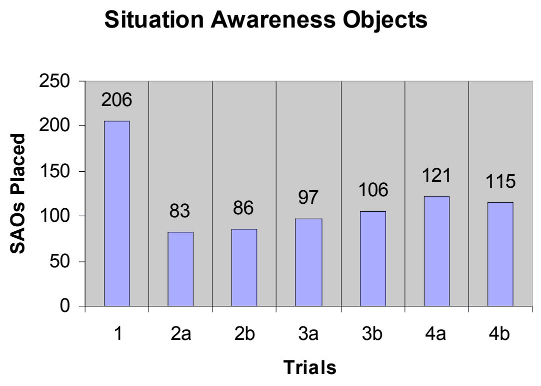


Figure 5. SAO count across the JUO-UR1 trials

Table 2 summarizes the SAO comments along the five situation dimensions and three situation awareness levels (perception 1A and B, comprehension, and projection). The SAO comments were categorized by a trained judge and independently verified. As an example, the comment, "Tank PLT 0755 - Tank platoon heading South from the airport," has three dimensional markers: (i.e., Entity – "Tank Platoon"; Time – 0755; Space – "heading South from the airport") and a first and second level SA (i.e., knowing that Red is a tank platoon and that Red is heading South from the airport). We also make a distinction between SA Level 1A and 1B. For the previous example, the SAO notes a "Tank Platoon," which is identified as level 1B because it is grouped (Platoon). The rest of our analyses are based on these counts.

Effects between the first and second week of trials 2, 3, and 4

As can be seen in Figure 6, there is a decrease in Level 2 SA from the first and second week, for trials 2a/2b, 3a/3b, and 4a/b. This may indicate that the "b" conditions are generally more difficult to identify than the "a" conditions. For example, in trial 2b, in addition to having Red being inactive, there is the additional factor of poor weather that players must contend with. In trial 3, a reduction of Red inventory seems to have a greater effect on Level 2 SA than signature reduction. Finally, in trial 4, the combination of no RF tags, ½ inventory, and signature reduction have a greater effect on Level 2 SA than no RF tags alone.

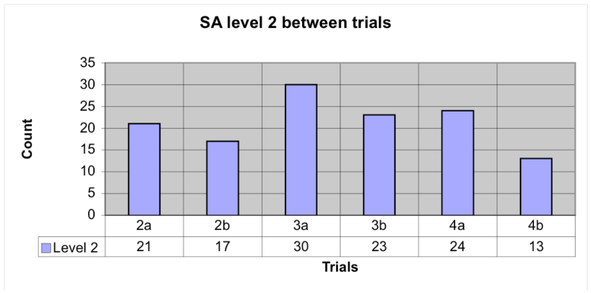


Figure 6. Level 2 SA for each experimental trial

Lower level SA and Situation Dimensions

In looking at Figure 7, it is apparent that entity information, followed by space and time, dominates the SAO comments. In contrast, there are relatively few comments that contain goal and causal information. The sheer amount of entity information reflects the fact the entity information was mentioned in almost every SAO. Additionally, spatial information tended to co-occur with temporal information.

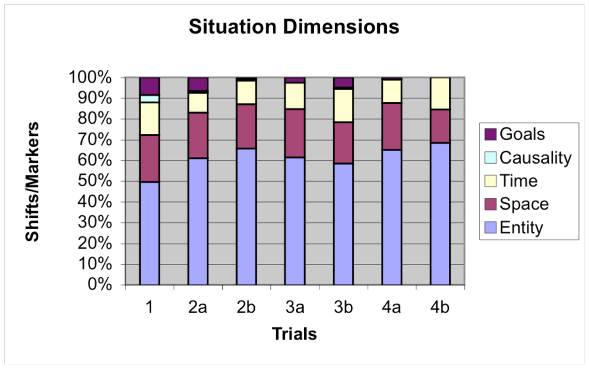


Figure 7. Situation dimensions for the seven trials

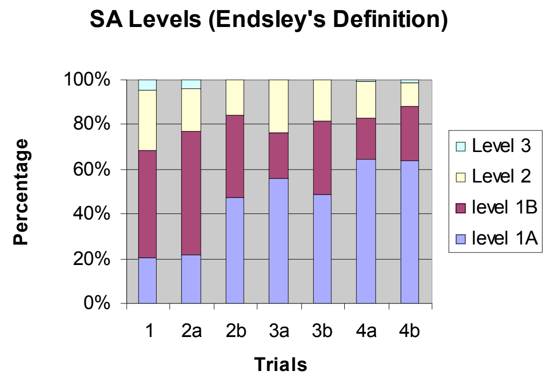


Figure 8. SA levels for the seven trials

Figure 8 shows that Level 1 SA. Similarly, levels 1a & 1b acounts for more than half of the SA levels recorded, and similar level 3 is only a small percentage of the total SA recorded, Figure 8.

Comparison Between Situation Dimensions and SA Levels

Next, we directly compare the situation model dimension counts across the SA Levels. We break this down in the following three figures 9-11. Figure 9 shows that SAOs that refer to entity information tend to be those that include SA information at the Level 1 perceptual level. Figure 10 shows that SAOs that include spatial and temporal information tend to be those that include SA information at the Level 2 comprehension level. Figure 11 did not provide clear cut evidence for a relationship between causal/goal information and SA information at the Level 3 projection level. However, we suspect that this is due to the fact that there are too few data points to make this a reliable comparison. Evidence for a relationship between situation model dimensions and levels of SA implies that efforts to automate situation awareness may consider the information provided by situation dimensions.

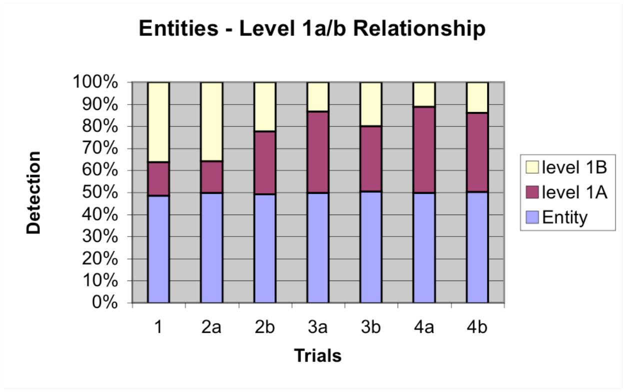


Figure 9. Entities – SA Level 1 Relationship

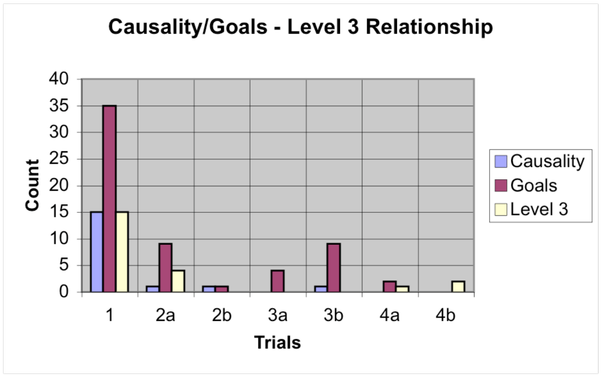


Figure 11. Causality/Goals – SA Level 3 Relationship

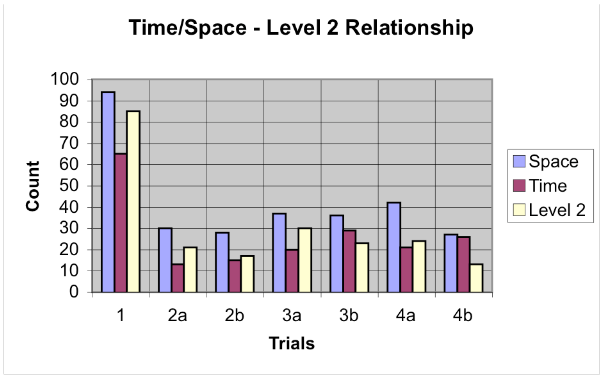


Figure 10. Time/Space – SA Level 2 Relationship

Table 2. SAO data collected for the seven trials of JUO

TRIAL	Entity	Space	Time	Causality	Goals	Level 1A	Level 1B	Level 2	Level 3
1	206	94	65	15	35	65	154	85	15
2a	83	30	13	1	9	24	60	21	4
2b	86	28	15	1	1	50	39	17	0
3a	97	37	20	0	4	72	26	30	0
3b	106	36	29	1	9	62	42	23	0
4a	121	42	21	0	2	95	27	24	1
4b	115	27	26	0	0	82	32	13	2

SUMMARY AND CONCLUSION

In summary, an analysis of SAOs recorded during a JUO Urban Resolve HITL experiment found evidence for a correspondence between levels of situation awareness and the situation model dimensions. Specifically, Level 1 SA comments included a relatively high proportion of spatial and temporal information, whereas Level 3 comments included information about the Red Force goals and intent. Our analysis is also consistent with previous observations that there tends to be more relatively information available about lower levels of SA.

This analysis yielded some interesting observations. Notably, causal information was lacking in players' comments. It is possible that players either did not ascribe causal relationships between events or they did notice causal relationships but did not record them. Determining causality is inherently more difficult than tracking entity locations and may have subsequently been less of a focus for the players. It does seem that increasing the ability to detect causal relationships between events would increase players' situation awareness.

Our approach differs from previous attempts to assess situation awareness in that it is based on entries players made during the experiment, rather than on observations of the players' activities both during and after the experiment. We believe that our approach is advantageous in that it has the potential to allow players to track their situation awareness online.

Future work will focus on addressing this possibility as well as modifying the manner in which data is recorded so that it is done more automatically. We are also interested in comparing our metrics of the players's SA against other methods that capture and analyze simulation groundtruth, e.g. the FAARS's data-collection effort (Graebener 2003) or the Cognitive Enabled ARCHitectures (CEARCH) project.

ACKNOWLEDGEMENTS

We wish to recognize our birddog, LTC Scott Shutzmeister who oversaw the completion of this paper. Special thanks go to Maston Gray, Jim Blank, Rae Denhke, Dr. Robert F. Lucas, Dan Davis, and Donna Brooks-Denhke for their support during various stages of this project. General Larry Budge (USA Ret.), Dr. Gabiel Radvansky, Dr. Andew Ceranowicz provided relevant comments on earlier versions of this paper. And lastly, special thanks go to NuGai Tran for providing the impetus to initiate and complete this project. AFRL Contracts F30602-02-C-0213 and FA875 0-05-2-0204.

REFERENCES

CEARCH. Cognitive Enable ARCHitures. Retrieved 24 June 2005 from http://cearch.east.isi.edu/

Ceranowicz, A. & Torpey, M. (2004). Adapting to Urban Warfare. Paper presented at the Interservice/Industry Training, Simulation, and Education Conference.

Endsley, M. R. (1998). Theoretical Underpinnings of Situation Awareness: A Critical Review. In M. R. Endsley & D. J. Garland (Eds.) Situational Awareness Analysis and Measurement. Mahwah, MJ: Lawrence Erlbaum.

Graebener, R., Rafuse, G., Miller, R. & Yao, K-T. (2003). The Road to Successful Joint Experimentation Starts at the Data Collection Trail. Paper presented at the Interservice/Industry Training, Simulation, and Education Conference.

Lutz, M. F., & Radvansky, G. A. (1997). The fate of completed goal information in narrative comprehension. Journal of Memory and Language, 36, 293-310.

Pew, R. W. & Mavor, A. S. (1998) (eds). Modeling human and organizational behavior. Washington D. C.; National Academy Press.

Prager, S., Cauble, K., Bakeman, D., Haes, S. & Goodman, G. (2004). Malls, Sprawl and Clutter: Realistic Terrain for Simulation of JUO. Interservice/Industry, Simulation, and Education Conference.

Speicher, D., & Wilbert, D. (2004). Simulating Urban Traffic in Support of the Joint Urban Operations Experiment. Interservice/Industry, Simulation, and Education Conference (I/ITSEC).

Speer, N. K. and Zacks, J. M. (2005). Temporal changes as event boundaries: Processing and memory consequences of narrative time shifts. Journal of Memory and Language, (additional info.).

Tran, J. J. Curiel, J. M. & Yao, K. (2004). An Interdisciplinary Approach to the Study of Battlefield Simulation Systems. Paper presented at the Interservice/Industry Training, Simulation and Education Conference.

Willams, R. & Tran, J.J. (2003). Supporting Distributed Simulation on Scalable Parallel Processor System. Paper presnted at the Interservice/Industry Training, Simulation and Education Conference.

Zwaan, R. A., and G.A. Radvansky, G. A. (1998). Situation models in language comprehension and memory. Psychological Bulletin, 123, 162-185.